√ ANNUAL STATUS REPORT

Error Control Techniques for Satellite and Space Communications

NASA Grant Number NAG5-557

Principal Investigator:
Daniel J. Costello, Jr.

October 1989

Summary of Progress

In this report, we will concentrate on two aspects of our work for NASA: the construction of multi-dimensional phase modulation trellis codes and a performance analysis of these codes. Appendix A contains the final version of a paper entitled "Trellis Coded Multi-Dimensional Phase Modulation" [1], earlier versions of which were included in our June 1988 and February 1987 reports. This paper, which will appear shortly in the *IEEE Transactions on Information Theory*, now contains a complete list of all the best trellis codes for use with phase modulation. L×MPSK signal constellations are included for M = 4, 8, and 16 and L = 1, 2, 3, and 4. Spectral efficiencies range from 1 bit/channel symbol (equivalent to rate 1/2 coded QPSK) to 3.75 bits/channel symbol (equivalent to rate 15/16 coded 16-PSK). The parity check polynomials, rotational invariance properties, free distance, path multiplicities, and coding gains are given for all codes. These codes are considered to be the best candidates for implementation of a high speed decoder for satellite transmission. In our next report, we will discuss the design of a hardware decoder for one of these codes, viz., the 16-state 3×8-PSK code with free distance 4.0 and coding gain 3.57 dB. This work is being conducted by Mr. Steven S. Pietrobon, a Ph.D. student supported by the grant.

Appendix B contains an exhaustive simulation study of the multi-dimensional phase modulation trellis codes. This study was motivated by the fact that coding gains quoted for almost all codes found in the literature are in fact only asymptotic coding gains, i.e., the coding gain at very high SNR or very low BER. These asymptotic coding gains can be obtained directly from a knowledge of the free distance of the code. On the other hand, real coding gains at BER's in the range of $10^{-2} - 10^{-6}$, where these codes are most likely to operate in a concatenated system, must be obtained by simulation. Our study goes far beyond anything previously published on this subject and forms the basis for a paper recently presented at the Allerton Conference on Communication, Control, and Computing [2]. This study was conducted by Mr. Lance C. Perez, a Ph.D. student supported by the grant.

Much other work has also been continuing since our last report. We will not discuss these in any detail here, but important results will be included in future reports. Other publications and presentations supported by the grant are listed in the references [3-9]. Copies of these as well as reprints of papers previously published with grant support are being sent to NASA under separate cover.

References

[1] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck, and D. J. Costello, Jr., "Trellis Coded Multi-Dimensional Phase Modulation", *IEEE Trans. Inform. Th.*, to appear.

[2] L. C. Perez and D. J. Costello, Jr., "On the Performance of Multi-Dimensional Phase Modulated Trellis Codes", Proc. Allerton Conf. on Commun., Cont., and Comput., Monticello, IL, September 1989.

[3] S. S. Pietrobon, D. J. Costello, Jr., and G. Ungerboeck, "A General Parity Check Equation for Rotationally Invariant Trellis Codes", 1989 IEEE Information Theory Workshop, Cornell University, Ithaca, NY, June 26, 1989.

[4] D. J. Costello, Jr., "An Introduction to Bandwidth Efficient Coding", Univ. of Illinois Coordinated Science Lab Seminar, Urbana, IL, July 21, 1989.

[5] D. J. Costello, Jr., "Bandwidth Efficient Concatenated Coding for Satellite Communications", Univ. of Illinois Coordinated Science Lab Seminar, Urbana, IL, July 21, 1989.

[6] S. Lin, D. J. Costello, Jr., W. H. Miller, J. C. Morakis, and W. B. Poland, Jr., "Bandwidth Efficient Coding for Satellite Communications", NASA Advanced Modulation and Coding Technology Conference, Cleveland, OH, June 22, 1989.

[7] D. J. Costello, Jr., C. Schlegel, and S. Lin, "Trellis Coded Modulation on Channels with Jamming and Impulse Noise", Proc. 1989 Conf. on Inform. Sciences & Systems, p. 210, Johns Hopkins Univ., Baltimore, MD, March 1989.

[8] E. Pryor, "Performance of Trellis Coded Modulation on non-AWGN Simulated Channels", M.S. Thesis, Univ. of Notre Dame, Notre Dame, IN, June 1989.

[9] N. Lebrec, "Multi-level Multi-dimensional Codes for 8-PSK and 16-PSK Modulation", M.S. Thesis, Univ. of Notre Dame, Notre Dame, IN, August 1989.

Appendix A
Trellis Coded Multi-Dimensional Phase Modulation

Trellis Coded Multi-Dimensional Phase Modulation[1]

Steven S. Pietrobon
Department of Electrical & Computer Engineering
University of Notre Dame, Notre Dame, IN 46556 U.S.A.
and School of Electronic Engineering
South Australian Institute of Technology
The Levels, P. O. Box 1, Ingle Farm S.A. 5098, Australia

Robert H. Deng
Institute of Systems Science, National University of Singapore
Kent Ridge, Singapore 0511
Formerly with the Department of Electrical and Computer Engineering
University of Notre Dame, Notre Dame, IN 46556

Alain Lafanechére
Schlumberger Industries
1 Rue Nieuport, 78141, Velizy, France
Formerly with the Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL 60616, U.S.A.

Gottfried Ungerboeck
IBM Zurich Research Laboratory
Säumerstrasse 4, CH-8803, Rüschlikon, Switzerland

Daniel J. Costello, Jr.
Department of Electrical and Computer Engineering
University of Notre Dame, Notre Dame, IN 46556, U.S.A.

September 20, 1989

Abstract

In this paper, trellis coded multi-dimensional MPSK modulation is investigated. A 2L-dimensional MPSK (L×MPSK) signal set is obtained by forming the Cartesian product of L 2-dimensional MPSK signal sets. A systematic approach to partitioning L×MPSK signal sets is used which is based on block coding. An encoder system approach is developed which incorporates the design of a differential precoder, a systematic convolutional encoder, and a signal set mapper. Trellis coded L×4PSK, L×8PSK, and L×16PSK modulation schemes are found for $1 \leq L \leq 4$ and a variety of code rates and decoder complexities, many of which are fully transparent to discrete phase rotations of the signal set. The new codes achieve asymptotic coding gains up to 5.85 dB.

[1]This work was supported in part by NASA Grant NAG5-557 and OTC (Australia) R&D Programme No. 4.

1 Introduction

Since the publication of the paper by Ungerboeck [1], Trellis Coded Modulation (TCM) has become a very active research area [2-13]. The basic idea of TCM is that by trellis coding onto an expanded signal set (relative to that needed for uncoded transmission), both power and bandwidth efficient communication can be achieved.

TCM can be classified into two basic types, the lattice type (e.g., M-PAM and M-QASK) and the constant amplitude type (e.g., MPSK). Constant amplitude modulation schemes have a lower power efficiency compared with lattice type modulation schemes but are more suitable for certain channels, e.g., satellite channels containing nonlinear amplifiers such as traveling wave tubes (TWT). Taylor and Chan [5] and Wilson et. al. [6] have studied the performance of trellis coded MPSK (TC-MPSK) modulation, in particular rate 2/3 TC-8PSK and rate 3/4 TC-16PSK, respectively, for various channel bandwidths and TWT operating points. Their results showed that TC-MPSK modulation schemes are quite robust under typical channel conditions.

In any TCM design, partitioning of the signal set into subsets with increasing minimum intra-subset distances plays a central role. It defines the signal mapping used by the modulator and provides a tight bound on the minimum free Euclidean distance (d_{free}) between code sequences. For lattice-type TCM, Calderbank and Sloane [10] have made the important observation that partitioning the signal set into subsets corresponds to partitioning a lattice into a sublattice and its cosets. Forney [13] has developed a method, called the squaring construction, of constructing higher-dimensional lattices from partitioned lower-dimensional lattices.

In this paper, we investigate a class of trellis coded multi-dimensional (multi-D) MPSK modulation schemes. Signals from a 2L-dimensional (2L-D) MPSK signal set (which we shall denote as L$\times$MPSK) are transmitted over a 2-D modulation channel by sending L consecutive signals of an MPSK signal set. Therefore, the L$\times$MPSK signal set is the Cartesian product of L 2-D MPSK signal sets. Trellis coded multi-D phase modulation (TC-L$\times$MPSK) provides us with a number of advantages that usually cannot be found with TC-MPSK: (i) flexibility in achieving a variety of fractional information rates, (ii) codes which are partially or totally transparent to discrete phase rotations of the signal set, (iii) suitability for use as inner codes in a concatenated coding system [14], due to their byte oriented nature, and (iv) higher decoder speeds resulting from the high rate codes used (rate $k/(k+1)$ with k up to 15 for some codes).

In Section 2, we introduce a block coding technique for partitioning L$\times$MPSK signal sets. Section 3 describes how the encoder system, comprising a differential precoder, a systematic convolutional encoder, and a multi-D signal set mapper, is obtained for the best codes found in a systematic code search. The signal sets are designed such that the codes can become transparent to integer multiples of $360^\circ/M$ rotations of the MPSK signal set. Also, due to the way in which they are mathematically constructed, a signal set mapper can be easily implemented by using basic logic gates and L bit binary adders. The systematic code search is based on maximizing d_{free} (and thus the asymptotic coding gain) as well as minimizing the number of nearest neighbors (N_{free}) for various degrees of phase transparency. TC-L$\times$4PSK, TC-L×8PSK, and TC-L×16PSK codes for $L = 1$ to 4 are found. For TC-L×8PSK and TC-L×16PSK, asymptotic coding gains up to 5.85 dB compared to an uncoded system are obtained. The TC-L×4PSK codes exhibit asymptotic coding gains up to 7.8 dB. Among the $L = 1$ codes listed are some new codes which have improvements in N_{free} and phase transparency compared to codes found previously [1, 4, 6, 15]. Viterbi decoding of TC-L×MPSK is also discussed, concentrating on maximum likelihood decoding of the parallel transitions within a code trellis.

2 Multi-D Signal Set Partitioning

In order to describe set partitioning we will start with the familiar partitioning of the 8PSK signal set. This is followed with an example of multi-D signal set partitioning using the 2×8PSK signal set. Generalizations will be gradually introduced, so that by the end of this section the reader should become thoroughly familiar with the concepts involved.

2.1 Partitioning the 8PSK Signal Set

In partitioning the 8PSK signal set, or 1×8PSK, we form a minimum squared subset distance (MSSD) chain of $\delta_0^2 = 0.586, \delta_1^2 = 2, \delta_2^2 = 4$, and $\delta_3^2 = \infty$ (assuming that the average signal energy is one). Figure 1 illustrates this partitioning, in which each subset is equally divided into two smaller subsets such that the MSSD in each smaller subset is maximized. Partitoning continues in this manner until we have eight subsets, each containing a single point, hence $\delta_3^2 = \infty$.

2.2 Partitioning 2×8PSK

A 2×8PSK signal set ($L = 2$) is illustrated in Figure 2. We use integers y_1 to indicate the first 8PSK point and y_2 for the second 8PSK point, where $y_1, y_2 \in \{0, 1, \ldots, 7\}$. Natural mapping is used to map the integer y_j into each complex valued 8PSK signal, i.e., $y_j \mapsto \exp[\sqrt{-1}y_j\pi/4]$, for $j = 1, 2$. We can also represent y_1 and y_2 in binary form as the vector $y_j = [y_j^2, y_j^1, y_j^0]$, with $y_j^i \in \{0, 1\}$, and where $y_j = 4y_j^2 + 2y_j^1 + y_j^0$, for $j = 1, 2$. That is, the least significant bit (*lsb*) of y_j corresponds to the right most bit and the most significant bit (msb) to the left most bit. We will use this convention throughout the paper.

To represent a 2×8PSK signal point we form the 2×3 binary matrix,

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} y_1^2 & y_1^1 & y_1^0 \\ y_2^2 & y_2^1 & y_2^0 \end{bmatrix}.$$

Since there are a total of six bits used to describe a signal point, the unpartitioned signal set (indicated by Ω^0) has a total of $2^6 = 64$ points. We also say that Ω^0 is at partition level $p = 0$. It can easily be seen that the MSSD at partition level $p = 0$ is $\Delta_0^2 = \delta_0^2, = 0.586$ (we use large Δ to indicate the MSSD's for $L > 1$ and small δ for $L = 1$). The next partition (at partition level $p = 1$) divides Ω^0 into two subsets of 32 points each. We call Ω^1 the subset that contains the all zero element (i.e., $y_1 = y_2 = 0$). The other subset of 32 points is its coset, labeled Ω^1 (1). In forming these two subsets, we would like their MSSD, Δ_1^2, to be larger than Δ_0^2. If this were not possible, then we should find a partitioning that leads to a maximum reduction in the number of nearest neighbors within the smaller subsets (i.e., the average number of signal points that are distance Δ_1^2 away from any point). In principle, the partitioning could be carried out in this heuristic manner.

A more efficient way of partitioning Δ^0 is to require the column vectors of $\mathbf{y}$, i.e., $\mathbf{y}^i = [y_1^i, y_2^i]^T$, for $0 \leq i \leq 2$, to be codewords in a block code. This representation using block codes is also known as multilevel coding (first described by Imai and Hirakawa [16] and later applied to QAM by Cusack [17]). To express this mathematically, we need to introduce some further notation. We define $\mathbf{C}_{m_i}$ as that block code which contains the column vectors $\mathbf{y}^i$, for $0 \leq i \leq 2$. Thus, $\mathbf{C}_{m_0}$ contains the least significant bits of y_1 and y_2, $\mathbf{C}_{m_1}$ contains the middle bits of y_1 and y_2, and so on. The actual value of m_i indicates which block code is being used. For $L = 2$ there are only three block codes that are of interest to us: $\mathbf{C}_0$, which is the (2,2) block code with Hamming distance $d_0 = 1$ (and code words $[00]^T, [01]^T, [11]^T$, and $[10]^T$), $\mathbf{C}_1$, which is the (2,1) block code with Hamming distance $d_1 = 2$ (and code words $[00]^T$ and $[11]^T$), and $\mathbf{C}_2$, which is the (2,0) block code having only one code word, $[00]^T$ and Hamming distance $d_2 = \infty$.

Also, since $\mathbf{C}_{m_i}$ denotes a block code with 2^{L-m_i} code words, we can write that the partition level p is the sum of all the m_i's that produce the subset Ω^p, i.e., $p = \sum_{i=0}^{2} m_i$. Since there are $I = \log_2 M$ bits needed for each MPSK point, p can range from 0 to IL (0 to 6 in this case). A shorthand way of writing which column vectors $\mathbf{y}^i$ belong to which block codes is $\Omega(\mathbf{C}_{m_2}, \mathbf{C}_{m_1}, \mathbf{C}_{m_0})$. Thus, we can write $\Omega^0 = \Omega(\mathbf{C}_0, \mathbf{C}_0, \mathbf{C}_0)$. Since $\mathbf{C}_0$ contains all possible length two binary vectors, then Ω^0 is generated.

To obtain the next partition (at level $p = 1$), we let $\Omega^1 = \Omega(\mathbf{C}_0, \mathbf{C}_0, \mathbf{C}_1)$. This partition satisfies our previous comments on partitioning. That is, there are only two code words in $\mathbf{C}_1$ (reducing the number of points to 32), and $\mathbf{C}_1$ contains the all zero code word. In partitioning, we also require the property that all the points in Ω^1 belong to Ω^0 (written as $\Omega^1 \subset \Omega^0$). For this example, since $\mathbf{C}_1 \subset \mathbf{C}_0$, this property is satisfied. This can be stated more generally as $\Omega^{p+1} \subset \Omega^p$, for $0 \leq p \leq IL - 1$. Thus, if we have two partition levels p and p', and $p' = p + 1$, then $\mathbf{C}_{m_i'} \subseteq \mathbf{C}_{m_i}$ for $0 \leq i \leq I - 1$.

The partition Ω^1 is equivalent to forcing the lsb's of y_1 and y_2 to be either both zero or both one. By inspection of Figure 2 we can thus see that $\Delta_1^2 = 2\delta_0^2 = 1.172$. In fact, we can use a more general expression which gives a lower bound on the MSSD. From [18,19] we have,

$$\Delta_p^2 \geq \min(\delta_{I-1}^2 d_{m_{I-1}}, \ldots, \delta_1^2 d_{m_1}, \delta_0^2 d_{m_0}), \tag{1}$$

where d_{m_i} is the Hamming distance of the code $\mathbf{C}_{m_i}$, for $0 \leq i \leq I - 1$. From (1), we obtain for 2×8PSK,

$$\Delta_p^2 \geq \min(4d_{m_2}, 2d_{m_1}, 0.586d_{m_0}). \tag{2}$$

For $p = 0$ and 1, we can see that (2) is satisfied with equality. In fact, due to the symmetry of the 8PSK signal set, (2) is an equality for all values of p. It can be seen that in partitioning Ω^0 into Ω^1 and its coset Ω^1 (1), we could have formed $\Omega(\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_0)$ or $\Omega(\mathbf{C}_1, \mathbf{C}_0, \mathbf{C}_0)$ instead of $\Omega(\mathbf{C}_0, \mathbf{C}_0, \mathbf{C}_1)$. However, both these other partitions have $\Delta_1^2 = 0.586$, and are therefore not good partitions, since we want Δ_1^2 to be as large as possible. This is because d_{free}^2 can be lower bounded by $2\Delta_1^2$ for many trellis codes [1].

Ignoring for the moment how the cosets are formed, we can partition Ω^1 into Ω^2 and its coset $\Delta^2(2)$, and so on. (The value within the brackets of the coset will be explained in Section 2.3.) Every time we partition, we want to make Δ_p^2 as large as possible. To do this we use the following rule. The $\mathbf{C}_{m_i}$ that we partition (into $\mathbf{C}_{m_i+1}$) from level p to level $p+1$ should be the i corresponding to the smallest $\delta^2 d_{m_i}$ at partition level p. If there are two or more $\delta_i^2 d_{m_i}$ that have the smallest value, we choose the one with the smallest i.

Note that once $\mathbf{C}_{m_i}$ has been partitioned to $\mathbf{C}_2$ (or $\mathbf{C}_{I-1}$ in general), then that particular block code cannot be further partitioned (since it contains only one code word). Table 1 illustrates the partitioning of the 2×8PSK signal set. The arrows show which $\mathbf{C}_{m_i}$'s are being partitioned as p is increased. The values of Δ_p^2 are also shown. Note that at $p = 3$, we have $\delta_i^2 d_{m_i} = 4$ for both $i = 1$ and 2. As indicated by the above rule, $i = 1$ is chosen to be partitioned to form Ω^4. Even though $\Delta_4^2 = \Delta_3^2 = 4$, partition level 4 is still useful for coding since the number of nearest neighbors for Ω^4 is less than for Ω^3. This will become more apparent when the actual codes are found.

The above rule usually works quite well. For $L = 3$, though, some of the best partitions do not follow this rule. Instead, we can allow a Δ_p^2 to be smaller than the rule proposes, in order to obtain a larger $\Delta_{p'}^2$ for some $p' > p$, than is possible by following the rule.

2.3 Formation of Cosets

Now consider partition level $p = 1$. We have shown that there are two subsets, namely Ω^1 and its coset $\Omega^1(1)$. To obtain $\Omega^1(1)$, we must look at how coset codes are derived from block codes. Recall that $\mathbf{C}_1$ is the (2,1) block code with Hamming distance $d_1 = 2$. The coset of this code, $\mathbf{C}_1(1)$, is formed by adding modulo-2 a non-zero code word that belongs to $\mathbf{C}_0$, but does not belong to $\mathbf{C}_1$ (called the *generator* τ^0), to all the code words in $\mathbf{C}_1$. We illustrate this with an example. $\mathbf{C}_0$ has code words $[0\ 0]^T$, $[0\ 1]^T$, $[1\ 0]^T$, and $[1\ 1]^T$ (remember that these code words correspond to column vectors of $\mathbf{y}$) and $\mathbf{C}_1$ has code words $[0\ 0]^T$ and $[1\ 1]^T$. Therefore, the generator τ^0 could equal $[0\ 1]^T$ or $[1\ 0]^T$. We arbitrarily choose $\tau^0 = [0\ 1]^T$. Thus, $\mathbf{C}_1(1) = \mathbf{C}_1 \oplus \tau^0 = \{[0\ 1]^T, [1\ 0]^T\}$. (In this paper the symbol $\oplus$ will be used to denote modulo-2 (exclusive-OR) arithmetic and + to denote integer or modulo-M, $M > 2$, arithmetic.) Note that if $\tau^0 = [1\ 0]^T$, the same coset vectors would have been found, except that they would have been in a different order. Also note that the Hamming distance between codewords in $\mathbf{C}_1(1)$ is equal to d_1.

We can also write a general expression for the cosets at partition level $p = 1$ as

$$\mathbf{C}_1(\zeta^0) = \mathbf{C}_1 \oplus \zeta^0\tau^0, \tag{3}$$

where $\zeta^0 \in \{0, 1\}$. Thus, when $\zeta^0 = 0$, we obtain $\mathbf{C}_1(0) \equiv \mathbf{C}_1$ and when $\zeta^0 = 1$, we obtain the coset of $\mathbf{C}_1, \mathbf{C}_1(1)$. In a similar way we can divide $\mathbf{C}_1$ into $\mathbf{C}_2$ and its coset $\mathbf{C}_2(2)$, and $\mathbf{C}_1(1)$ into cosets $\mathbf{C}_2(1)$ and $\mathbf{C}_2(3)$. Figure 3 gives an illustration of this partition. For the second generator, there is only one choice, i.e., $\tau^1 = [1\ 1]^T$. The general expression for the cosets at partition level $p = 2$ becomes

$$\mathbf{C}_2(2\zeta^1 + \zeta^0) \;=\; \mathbf{C}_2 \oplus \zeta^1\tau^1 \oplus \zeta^0\tau^0$$

$$= \zeta^1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \oplus \zeta^0 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad (4)$$

where $\mathbf{C}_2$ is the all zero vector and $\zeta^m \in \{0,1\}$, for $0 \leq m \leq 1$. We also note that $\mathbf{C}_2 \subset \mathbf{C}_1 \subset \mathbf{C}_0$ and that $\tau^m \in \mathbf{C}_m$, but that $\tau^m \notin \mathbf{C}_{m+1}$, for $0 \leq m \leq 1$.

Since we have shown how the cosets of $\mathbf{C}_m$ are formed, we can now show how the cosets of Ω^p are formed. We start with the simplest case, the single coset of Ω^1 namely $\Omega^1(1)$. In the same way as the block codes are partitioned, we must find a 2×3 matrix that belongs to Ω^0 but does not belong to Ω^1. This is called the *generator* of Ω^1 and is labeled $\mathbf{t}^0$. Since $\mathbf{C}_{m_0}$ is partitioned in going from Ω^0 to Ω^1, this implies that $\mathbf{t}^0 = [\mathbf{0}, \mathbf{0}, \tau^0]$, where $\mathbf{0}$ is the all zero vector $[0\ 0]^T$, i.e.,

$$\mathbf{t}^0 = \begin{bmatrix} 0\,0\,0 \\ 0\,0\,1 \end{bmatrix}.$$

An alternate notation for $\mathbf{t}^0$ (using the symbol t^0), is to treat $\mathbf{t}^0$ as if it represented two integer values, y_1 and y_2. Thus, $\mathbf{t}^0$ in integer form is $t^0 = [0\ 1]^T$.

To form the coset $\Omega^1(1)$, all that is required is to add $\mathbf{t}^0$ modulo-2 to all the signal points in Ω^1. We write this as

$$\Omega^1(z^0) = \Omega^1 \oplus z^0\mathbf{t}^0, \quad (5)$$

where $z^0 \in \{0,1\}$ indicates which of the two subsets is being selected. We can see that in coset $\Omega^1(1)$, the lsb's of y_1 and y_2 are either 0 and 1 or 1 and 0, respectively. Thus this coset has the same MSSD as Ω^1, i.e., $\Delta_1^2 = 1.172$. Alternately, t^0 can be added modulo-M (modulo-8 in this case) to the signal points in Ω^1. With modulo-8 arithmetic, the lsb's of y_1 and y_2 are still added modulo-2, but the lsb's now produce carries which affect the middle and most significant bits. This is denoted as

$$\Omega^1(z^0) = \Omega^1 + z^0 t^0 \pmod 8. \quad (6)$$

For example, a signal $y = [1\ 3]^T$ (where $y = [y_1 y_2]^T$) in Ω^1 becomes $[1\ 2^T]$ with modulo-2 addition of $\mathbf{t}^0$ to $\mathbf{y}$ or $[1\ 4]^T$ with modulo-8 addition of t^0 to y. Using either type of arithmetic, we still obtain the required partition, although the ordering of signal points within each coset is different. In constructing rotationally invariant trellis codes, we will find that there is a distinct advantage to using modulo-M arithmetic over modulo-2 arithmetic.

Continuing with the set partitioning, it should be obvious that the next generator is $t^1 = [1\ 1]^T$. From Table 1, we see that t^1 corresponds to the generator of $\mathbf{C}_1$. The expression for the cosets of Ω^2 is

$$\Omega^2(2z^1 + z^0) = \Omega^2 + z^1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + z^0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod 8, \quad (7)$$

where $z^i \in \{0,1\}$, for $0 \leq i \leq 1$. For partition level $p = 3$, we choose $t^2 = [0\ 2]^T$, with $z^2 \in \{0,1\}$ used to select t^2. Continuing in the same way, we can partition the signal set until we obtain only a single (4-D) signal point. Thus we can form the equation (using the generators from Table 1).

$$y(z) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \Omega^6(z)$$

$$= z^5\begin{bmatrix}4\\4\end{bmatrix} + z^4\begin{bmatrix}0\\4\end{bmatrix} + z^3\begin{bmatrix}2\\2\end{bmatrix} + z^2\begin{bmatrix}0\\2\end{bmatrix} + z^1\begin{bmatrix}1\\1\end{bmatrix} + z^0\begin{bmatrix}0\\1\end{bmatrix} \pmod 8, \quad (8)$$

where $z = \sum_{i=0}^{5} 2^i z^i$, with $z^i \in \{0,1\}$, for $0 \le i \le 5$, and $y(z)$ gives the integer representations of the two 8PSK signal points. The signal set mapping given by z can now be directly used by a convolutional encoder. Since y_1 and y_2 can be described in terms of z, the signal set mapper can be implemented using simple logic circuits (exclusive-OR circuits for modulo-2 addition and binary adders for modulo-M addition). Alternatively, since z can be represented with only six bits, one can use a small ROM. Figure 4 illustrates two possible signal set mappers for 2×8PSK. Figure 4(a) shows a mapper using modulo-2 arithmetic, and Figure 4(b) shows a mapper using modulo-8 arithmetic.

In general, we can write (8) as

$$y(z) = \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = \Omega^{IL}(z) = \sum_{i=0}^{IL-1} z^i t^i, \quad (9)$$

where $z = \sum_{i=0}^{IL-1} 2^i z^i$, with $z^i \in \{0,1\}$, for $0 \le i \le IL-1$. The addition in (9) is not specified, but may be modulo-2 (using the binary matrix generators), modulo-M (using the integer generators), or a combination of modulo-2 and modulo-M. Figure 5 illustrates the partitioning of Ω^0 into Ω^3 and its cosets $\Omega^3(4z^2 + 2z^1 + z^0)$ for the 2×8PSK signal set using modulo-8 addition.

2.4 Partitioning 3×MPSK and 4×MPSK Signal Sets

In a similar fashion to 2×8PSK, to partition L×8PSK (for $L > 2$) requires the partitioning of length $L > 2$ block codes. We again look for partitions that have an increasing Hamming distance. For $L = 3$, there are two partitions that are interesting.

The first partition has Hamming distances $d_0 = 1, d_1^1 = 2$, $d_2^1 = 2$, and $d_3 = \infty$. These Hamming distances correspond to the (3,3), (3,2), (3,1), and (3,0) block codes $\mathbf{C}_0$, $\mathbf{C}_1^1$, $\mathbf{C}_2^1$, and $\mathbf{C}_3$, respectively, where $\mathbf{C}_3 \subset \mathbf{C}_2^1 \subset \mathbf{C}_1^1 \subset \mathbf{C}_0$. Table 2(a) gives the three generators, τ_1^0, τ_1^1, and τ_1^2, that were chosen, along with the Hamming distances (d_m) and the number of nearest neighbors (N_m) at each partition level m. The choice was not completely arbitrary, since one of the generators must be the all ones vector (which in this case is τ_1^0). The reason for this will be explained in Section 3.

It is interesting to note that the generator matrix for these block codes can be formed from the generators. In general, a generator matrix $\mathbf{G}_m$ for an $(L, L-m)$ block code $\mathbf{C}_m$, for $0 \le m \le L-1$, can be formed from the generators τ^m to τ^{L-1} , i.e., $\mathbf{G}_m = [\tau^m, \tau^{m+1}, \ldots, \tau^{L-1}]^T$. For example, for the $L = 3$ block codes given in Table 2(a),

$$\mathbf{G}_0^1 = \begin{bmatrix} 1\,1\,1 \\ 1\,1\,0 \\ 0\,1\,1 \end{bmatrix}, \mathbf{G}_1^1 = \begin{bmatrix} 1\,1\,0 \\ 0\,1\,1 \end{bmatrix}, \mathbf{G}_2^1 = [011].$$

For the other $L = 3$ partition, we have $d_0 = 1, d_1^2 = 1, d_2^2 = 3$, and $d_3 = \infty$. These distances correspond to block codes $\mathbf{C}_0, \mathbf{C}_1^2, \mathbf{C}_2^2$, and $\mathbf{C}_3$, where $\mathbf{C}_3 \subset \mathbf{C}_2^2 \subset \mathbf{C}_1^2 \subset \mathbf{C}_0$. Table

2(b) shows the generators for these codes. Note that τ_2^2 is the all ones vector in this case. The advantage of this partition is that $d_2^2 = 3$ is larger than $d_2^1 = 2$. However, $d_1^2 = 1$ is less than $d_1^1 = 2$.

The partitions of 3×8PSK that will be useful for trellis coding are given in Table 3. Table 3(a) corresponds to the first partition where we try to maximize Δ_p^2 at each partition level. In Tables 3(b) and 3(c), the second set of block codes are used to increase Δ_2^2 to 1.757 while Δ_1^2 decreases to 0.586. In Table 3(c), Δ_6^2 increases to 6.0 and Δ_4^2 decreases to 2.0. Note how $\Delta_6^2 = 6.0$ is obtained in Table 3(c). At $p = 4$ we have $\Delta_4^2 = \min(4.0, 2.0, \infty)$ and at the next partition level, $\Delta_5^2 = \min(4.0, 6.0, \infty) = 4.0$. Now $\mathbf{C}_{m_2}$ is partitioned to give $\Delta_6^2 = \min(8.0, 6.0, \infty) = 6.0$. In the next level, we partition $\mathbf{C}_{m_1}$ to obtain $\Delta_7^2 = 8.0$. In Section 3, the reasons why these latter two partitions are used will be seen more clearly.

For $L = 4$ there is only one good way to partition length 4 block codes. Table 2(c) gives a summary of the basic parameters. Using Table 2(c), we can partition the 4×8PSK signal set as shown in Table 4.

For L×4PSK and L×16PSK we obtain from (1) that,

$$\Delta_p^2 \geq \min(4d_{m_1}, 2d_{m_0}), \tag{10a}$$

$$\Delta_p^2 \geq \min(4d_{m_3}, 2d_{m_2}, 0.586d_{m_1}, 0.152d_{m_0}), \tag{10b}$$

respectively, where $p = \sum_{i=0}^{I-1} m_i (I = 2$ for (10a) and $I = 4$ for (10b)). In a similar fashion to L×8PSK, the signal set partitions can be obtained for $L = 2$ to 4. Tables 5, 6, and 7 give a summary of the partitions for L×4PSK, L×8PSK, and L×16PSK, respectively.

2.5 Larger Dimensional MPSK Signal Sets and the Squaring Construction

One way to obtain larger dimensional MPSK signal sets is to take an L×MPSK signal set partition (with its corresponding MSSD's relabeled as δ_i^2, for $0 \leq i \leq IL$) and form a $2LL'$ dimensional MPSK signal set which we label as $L' \times L\times$ MPSK. Thus if we have a 2×8PSK signal set, the MSSD's $\Delta_p^2, 0 \leq p \leq 6L'$, for $L' \times 2\times$ 8PSK are given by

$$\Delta_p^2 \geq \min(8d_{m_5}, 4d_{m_4}, 4d_{m_3}, 2d_{m_2}, 1.172d_{m_1}, 0.586d_{m_0}), \tag{11}$$

where the d_{m_i}'s are the Hamming distances of $(L', L'-m_i)$ block codes. If $L' = 2$ we can form the 2×2×8PSK signal set, which is equivalent to the 4×8PSK signal set. Table 8 illustrates this partitioning. Note that the MSSD's obtained are exactly the same as those found with the 4×8PSK partitioning given in Table 4. Figure 6 shows a block diagram of a signal set mapper for the partition of 2×2×8PSK. The function T_1 corresponds to the mapping given by the generators in Table 8 and T_2 to the generators in Table 4.

For $L' = 2$, the above method of obtaining larger dimensional MPSK is essentially equivalent to the squaring or two-construction described by Forney [13]. The cubing or three-construction corresponds to $L' = 3$. One can continue squaring or cubing various multi-D signal sets in an iterative fashion to obtain many larger dimensional signal sets. If we desire an L×MPSK signal set, all that is required is to factor L to determine which constructions are needed. For example, if $L = 24$, we could factor this into a $2 \times 2 \times 2 \times 3$×8PSK signal set.

If L is a prime number, then the appropriate length L block codes and their corresponding generators must be found.

Table 9 gives the generators for $L = 5$ and 7. Also given are the Hamming distances and the number of nearest neighbors for each length L block code. Note that there are three different partitions for $L = 5$ and four different partitions for $L = 7$. This seems to suggest that the number of useful partitions increases by one for each successive prime number. Thus, $L = 11$ is expected to have five useful partitions, and so on. These partitions were constructed by hand and probably represent the practical limit of hand constructions. For $L = 11$ and above, an algorithmic or mathematical method is required. In forming each partition, we have tried to maximize the Hamming distance and minimize the number of nearest neighbors. For example, the type IV partition maximizes the Hamming distance and minimizes the number of nearest neighbors for the (7,4) block code while the type III partition maximizes the Hamming distance and minimizes the number of nearest neighbors for the (7,3) and (7,2) block codes.

For larger dimensions, these methods may produce block codes which do not have the largest possible minimum distance. For example, the largest Hamming distance that can be obtained for the (24,12) coset code is six. However, the (24,12) Golay code has a Hamming distance of eight. For L = 2, 3, and 4, the block codes are relatively simple. Thus, we are fairly certain that the best partitions for these L×MPSK signal sets have been found.

3 Trellis Coded Multi-D MPSK Design

This section describes how convolutional codes are constructed for the L×MPSK signal sets described previously. We first show how to construct signal sets which have good phase rotation properties. Following this, a method used to find good convolutional codes based on the parity check equations is presented.

3.1 Construction of Signal Sets

Equation (9) can be used to describe a signal point in an L×MPSK signal set. The number of bits z^j needed to describe each signal point is IL. If the lsb is used for coding, we can form a rate $(IL-1)/IL$ code. A more convenient measure of rate is to use the average number of information bits transmitted during each 2-D signal period (T). This is called the *effective rate* of the code, $R_{eff} = (IL-1)/L$ (bit/T). The unit bit/s/Hz can also be used (for the actual bandwidth efficiency), but this assumes that perfect Nyquist filtering is used in the receive and transmit filters. Since this is not the case in many practical systems, we make a distinction between the units bit/T and bit/s/Hz.

Other rates can be achieved by setting the q lsb's of the mapping to 0. We do this to insure that the MSSD's are as large as possible, so that the best codes can be found. In this case (9) can be rewritten as

$$y^q(z) = \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} = \sum_{j=q}^{IL-1} z^{j-q} t^j, \tag{12}$$

for $0 \leq z \leq 2^{IL-q-1} - 1, 0 \leq q \leq L-1$, and where $y^q(z)$ represents a point z in an L×MPSK signal set such that the first q bits of (9) are 0. As before, we do not restrict the type of addition that is used. We now let $\mathbf{z} = [z^{IL-q-1}, \ldots, z^1, z^0]$, where $\mathbf{z}$ is the binary representation of z, and the lsb of z is always the coding bit. This notation insures that the parity check equations of a convolutional code can always be expressed in terms of the lsb's of z without depending on the type of signal set used or its partitioning. From (12), codes with effective rates $R_{eff} = (IL - q - 1)/L$ can be formed. An upper limit of $q = L - 1$ is set because for $q \geq L$ the signal set is partitioned such that $d_{m_0} = \infty$, i.e., an $M/2^j$ -PSK, for $j \geq 1$, signal set is being used (one exception is the 4×8PSK signal set (Table 4) where $d_{m_0} = 4$ for $q = L$). The MSSD's range from Δ_q^2 to Δ_{IL}^2 and the uncoded minimum squared Euclidean distance (MSED) is Δ_{q+1}^2, since uncoded transmission uses only half as many signals as coded transmission.

Example 3.1

We can form a rate 4/5 code with an effective rate of 2.0 bit/T from a 2×8PSK ($L = 2, I = 3$) signal set with $q = 1$. Then

$$y^1(z) = z^4 \begin{bmatrix} 4 \\ 4 \end{bmatrix} + z^3 \begin{bmatrix} 0 \\ 4 \end{bmatrix} + z^2 \begin{bmatrix} 2 \\ 2 \end{bmatrix} + z^1 \begin{bmatrix} 0 \\ 2 \end{bmatrix} + z^0 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \pmod 8.$$

The uncoded MSED is $\Delta_2^2 = 2.0$, which is the same as uncoded 4PSK.

3.2 Effect of a 360°/M Phase Rotation on a Multi-D MPSK Signal Set

Using modulo-M arithmetic in (12), multi-D signal sets can be constructed such that there are at most I bits in $\mathbf{z}$ affected by a signal set rotation of $\Psi \triangleq 360°/M$. For 4PSK, 8PSK, and 16PSK, this corresponds to rotations of 90°, 45°, and 22.5°, respectively. Initially, we consider all possible mapped bits, i.e., $q = 0$.

Consider that a 1×MPSK signal set has been rotated by Ψ. Since we are using natural mapping, the integer representation of the rotated signal point is $y_r = y + 1 \pmod{M}$, where y is the integer representation of the signal point before rotation. If y is in binary notation, then

$$y_r^0 = y^0 \oplus 1 = \overline{y^0}, \tag{13a}$$
$$y_r^1 = y^1 \oplus y^0, \tag{13b}$$
$$y_r^2 = y^2 \oplus y^0 \cdot y^1, \tag{13c}$$
$$\vdots$$

If there are $I = \log_2 M$ bits in a signal set, then we see from (13) that all I bits are affected by a phase rotation of Ψ.

Consider the 2×8PSK signal set, with the mapping given by (8). The phase rotation equations of this mapping can be determined as follows. From (8), the signal outputs can be written in terms of z as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (4z^5 + 2z^3 + z^1) \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (4z^4 + 2z^2 + z^0) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \pmod 8. \tag{14}$$

be written in terms of z as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (4z^5 + 2z^3 + z^1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (4z^4 + 2z^2 + z^0)\begin{bmatrix} 0 \\ 1 \end{bmatrix} (\text{mod } 8). \tag{14}$$

After a 45° phase rotation we have $y_{j,r} = y_j + 1(\text{mod } 8)$, for $j = 1, 2$. From (14), we can form the following phase rotation equations,

$$\begin{bmatrix} y_{1,r} \\ y_{2,r} \end{bmatrix} = (4z^5 + 2z^3 + z^1 + 1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (4z^4 + 2z^2 + z^0)\begin{bmatrix} 0 \\ 1 \end{bmatrix} (\text{mod } 8).$$

Note that a 1 is added to the term whose coset is $[1\ 1]^T$. Hence this term "absorbs" the effect of the phase rotation, leaving the remaining term unaffected. As can be seen, bits z^5, z^3, and z^1 are affected in a manner similar to y^2, y^1, and y^0 in (13), and bits z^4, z^2, and z^0 are unaffected by the phase rotation. Thus, we can form the phase rotation equations

$$\begin{array}{lll} z_r^0 = z^0, & z_r^2 = z^2, & z_r^4 = z^4, \\ z_r^1 = z^1 \oplus 1, & z_4^3 = z^3 \oplus z^1, & z_r^5 = z^5 \oplus z^1 \cdot z^3. \end{array} \tag{15}$$

If the signal set had been constructed using modulo-2 addition (instead of modulo-8), only z^0 would have remained unchanged by a 45° phase rotation. Using general notation, we can express (14) as

$$\begin{aligned} \begin{bmatrix} y_1 \\ \vdots \\ y_L \end{bmatrix} &= \left(2^{I-1} z^{p_{I-1}} + \cdots + 2z^{p_1} + z^{p_0}\right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\ &+ \ 2^{I-1}\{g_{I-1}\} + \cdots + 2\{g_1\} + \{g_0\} \ \ (\text{mod } M), \end{aligned} \tag{16}$$

where p_j, for $0 \leq j \leq I - 1$, corresponds to those partition levels where t^p equals the vector $[2^j, 2^j, \ldots, 2^j]^T$. The term g, for $0 < j < I - 1$, corresponds to those remaining terms that have at least one (but not all) component in t^p with value 2^j. For (14) we would have $p_0 = 1, p_1 = 3$, and $p_2 = 5$. These values of p_j are given for all the signal set partitions shown in Tables 5-7. We can now write the phase rotation equations as

$$z_r^{p_0} = z^{p_0} \oplus 1, z_r^{p_1} = z^{p_1} \oplus z^{p_0}, z_r^{p_2} = z^{p_2} \oplus z^{p_0} \cdot z^{p_1}, \ldots \tag{17}$$

and for all other partition levels, $z_r^p = z^p$.

For $L = 2$, there is only one term in each g_j. However, for $L \geq 3$, there are two or more terms in each g_j. Since the terms in g_j do not contribute to the phase rotational properties of the signal mapping, these terms can be added modulo-2 before being added modulo-M to the other terms. This is best illustrated with an example. For the 3×8PSK (I) signal set in Table 3(a), we have the following mapping equation:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = z^8 \begin{bmatrix} 0 \\ 4 \\ 4 \end{bmatrix} + z^7 \begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} + z^6 \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} + z^5 \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} + z^4 \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$$

$$\begin{aligned}
&+ \quad z^3\begin{bmatrix}2\\2\\2\end{bmatrix} + z^2\begin{bmatrix}0\\1\\1\end{bmatrix} + z^1\begin{bmatrix}1\\1\\0\end{bmatrix} + z^0\begin{bmatrix}1\\1\\1\end{bmatrix} \\
&= \quad (4z^6+2z^3+z^0)\begin{bmatrix}1\\1\\1\end{bmatrix} + 4\left\{z^8\begin{bmatrix}0\\1\\1\end{bmatrix} \oplus z^7\begin{bmatrix}1\\1\\0\end{bmatrix}\right\} \\
&+ \quad 2\left\{z^5\begin{bmatrix}0\\1\\1\end{bmatrix} \oplus z^4\begin{bmatrix}1\\1\\0\end{bmatrix}\right\} + \left\{z^2\begin{bmatrix}0\\1\\1\end{bmatrix} \oplus z^1\begin{bmatrix}1\\1\\0\end{bmatrix}\right\} \pmod 8 \\
&= \quad (4z^6+2z^3+z^0)\begin{bmatrix}1\\1\\1\end{bmatrix} + 4\begin{bmatrix}z^7\\z^8\oplus z^7\\z^8\end{bmatrix} + 2\begin{bmatrix}z^4\\z^5\oplus z^4\\z^5\end{bmatrix} + \begin{bmatrix}z^1\\z^2\oplus z^1\\z^2\end{bmatrix} \pmod 8.
\end{aligned}$$

The reason for this combination of modulo-2 and modulo-M arithmetic is that it reduces the number of logic circuits required in a signal set mapper. For small IL, it may be simpler to use ROM's for signal set mapping, but for large IL this dual addition becomes preferable. Figure 7 gives a block diagram of the three 3×8PSK signal set mappers and Figure 8 illustrates the mapper for 4×8PSK. This combination of modulo-2 and modulo-M addition has no effect on the MSSD's (at least for $L \leq 4$). In a similar manner, we can also obtain the signal set mappers for L×4PSK and L×16PSK.

Due to the phase rotational properties and simplified hardware that the combined modulo-2 and modulo-M mapping allows, these are the signal sets that are used to find all the trellis codes in this paper.

We have shown that for $q = 0$, the bits that are affected by a phase rotation of Ψ are z^{p_j}, for $0 \leq j \leq I-1$. For $q > 0$ the bits that are affected are $z^{p_j - q}$, for $0 \leq j \leq I-1$. However, depending on the signal set, $p_j - q$ for some j may be less than zero. If this is true, the minimum phase transparency is $2^{d'}\Psi$, where d' is the number of terms $p_j - q$ that are less than zero, and the number of bits that are affected by a $2^{d'}\Psi$ phase rotation is $s' = I - d'$. For example, the 3×8PSK signal set in Table 3(a) has $p_0 = 0, p_1 = 3$, and $p_2 = 6$. Thus if $q = 1$, then $p_0 - q = -1$, which is less than zero, implying that $d' = 1$, and thus only $s' = I - d' = 2$ bits are affected by a $2\Psi = 90°$ phase rotation. (A phase rotation of $\Psi = 45°$ of this signal set produces its coset.)

Fortunately, for the codes and signal sets considered in this paper, the above complication does not occur. This is partly due to the fact that for many signal sets with $q = 0$, the first $L-1$ lsb's are not affected by a phase rotation of Ψ. Since we consider only signal sets with $0 \leq q \leq L-1, d' = 0$ in these cases. For those signal sets where this is not true (e.g., in some 3×MPSK signal sets), it has been found that the convolutional codes produced are inferior (in either d_{free} or number of nearest neighbors) to an alternative signal set with $d' = 0$.

When a signal set is combined with a convolutional encoder we must consider the effect of rotating coded sequences. A similar result to above is obtained so that, depending on the code and the signal set, the signal set can be rotated in multiples of $2^d\Psi$ and still produce valid code sequences (where d defines the degree of transparency). The actual determination of d is described in Section 3.4. The number of bits that are affected by a $2^d\Psi$ phase rotation is $s = I - d$.

For $0 \leq q \leq L-1$, the actual bits that are affected by a phase rotation of Ψ are z^{b_j}, where $b_j = p_j - q$, for $0 \leq j \leq I-1$. More generally, the bits that are affected by a phase rotation of $2^d\Psi$ are z^{c_j}, where $c_j = p_{j+d} - q$, for $0 \leq j \leq s-1$. These two separate notations (b_j and c_j) are used because the determination of d depends on b_j, as will be shown in Section 3.4.

3.3 The General Encoder System

From the above information we can now construct a suitable encoder system, as illustrated in Figure 9. The general encoder system consists of five sections. These sections are the differential precoder, the binary convolutional encoder, the multi-D signal set mapper, the parallel to serial converter, and the 2-D signal set mapper. The convolutional encoder is assumed to be in feedback systematic form, as in [1]. That is, $z^j(D) = x^j(D)$ for $1 \leq j \leq k$, where D is the delay operator and polynomial notation is used. The parity sequence, $z^0(D)$, will be some function of itself and the $x^j(D)$, for $1 \leq j \leq k$. The parity check equation of an encoder describes the relationship in time of the encoded bit streams. It is a very useful and efficient means of describing high rate convolutional codes, since it represents the input/output encoder relationships in a single equation. For an $R = k/(k+1)$ code, the parity check equation is

$$H^{\tilde{k}}(D)z^{\tilde{k}}(D) \oplus \cdots \oplus H^1(D)z^1(D) \oplus H^0(D)z^0(D) = 0(D), \tag{18}$$

where $\tilde{k}, 1 \leq \tilde{k} \leq k$, is the number of input sequences that are checked by the encoder, $H^j(D)$, for $0 \leq j \leq \tilde{k}$, is the parity check polynomial of $z^j(D)$, and $0(D)$ is the all zero sequence.

Since the encoder is systematic, the differential precoder codes only those bits which are affected by a phase rotation. The input bits into the encoder which are precoded are denoted $w^{c_0}, w^{c_1}, \ldots, w^{c_{s-1}}$. If $c_0 = 0$, we replace w^0 (which does not exist) by z^0, as shown in Figure 9 by the dashed line (a different precoder must then be used). For example, an encoder for a rate 8/9 code which uses the 3×8PSK (I) signal set given in Table 3(a) may (depending on the phase transparency) need this modification. This is because this signal set has $b_0 = 0$, and thus if the code has $d = 0$, then z^0 will need to be precoded. Figure 10 illustrates the two types of precoders. Note that the storage elements have a delay of LT. Figure 10(a) illustrates the precoder with $c_0 > 0$, where there are s inputs that are precoded. The basic component of the precoder is the modulo-2^s adder. For most codes this is the precoder to be used. For the bits that are not precoded, $x^i = w^i$, for $i \neq c_j$.

Figure 10(b) shows the other case, where $c_0 = 0$ and $s-1$ input bits are precoded (the other precoded bit is z^0). The adder circuit for this case is different from Figure 10(a), i.e., it is not a modulo-2^s adder. The Appendix gives the equations for the differential encoder and decoder (for both cases) and an explanation of how these circuits work.

At this point, we summarize the notation and indicate the limits on the parameters used in the search for good codes. For a rate $(IL-q-1)/(IL-q)$ code,

I = no. of bits in each 2-D signal ($2 \leq I \leq 4$),
$M = 2^I$ = no. of signal points in each 2-D signal set,

L = no. of 2-D signal sets $(1 \leq L \leq 4)$,
p = partition level of signal set $(0 \leq p \leq IL)$,
q = the partition level p where mapping begins $(0 \leq q \leq L-1)$,
z = signal set mapping parameter $(0 \leq z \leq 2^{p-q}-1)$,
k = $IL - q - 1$ = no. of input bits to encoder,
$\tilde{k}$ = no. of bits checked by encoder $(1 \leq \tilde{k} \leq k)$,
$\Psi = 360°/M$ = minimum phase transparency with $q = 0$,
p_j = the bits z^{p_j} affected by a Ψ phase rotation with $q = 0$,
d = degree of phase transparency ($2^d\Psi$, for $0 \leq d \leq I$),
s = I-d = no. of bits in z affected by a $2^d\Psi$ phase rotation,
$c_j = p_{j+d} - q$ = the bits z^{c_j} affected by a $2^d\Psi$ phase rotation.

There are two types of systematic convolutional encoders that can be constructed. Before proceeding with the description of these encoders, we return to the parity check equation given in (18). As in [1], we define the *constraint length* ν to be the maximum degree of all the parity check polynomials $H^j(D)$, for $0 \leq j \leq \tilde{k}$. For $\tilde{k} < j < k, H^j(D) = 0$, since the bits corresponding to these polynomials are not checked by the encoder. The parity check polynomials are of the form

$$H^j(D) = 0 \oplus h^j_{\nu-1}D^{\nu-1} \oplus \cdots \oplus h^j_1 D \oplus h^j_0, 1 \leq j \leq \tilde{k}, \tag{19a}$$

$$H^0(D) = D^\nu \oplus h^0_{\nu-1}D^{\nu-1} \oplus \cdots \oplus h^0_1 D \oplus 1. \tag{19b}$$

If $\tilde{k} < \nu$, we let $h^j_0 = 0$, for $1 \leq j \leq \tilde{k}$. This insures that the squared Euclidean distance (SED) between paths in a trellis leaving or entering a state is at least Δ^2_{q+1}. Thus all codes in this class have a MSED between all possible non-parallel coded sequences of at least $2\Delta^2_{q+1}$. The parallel transitions provide an upper bound on the d_{free} of a code. A theoretical justification for constructing codes in this manner can be found in [20] where it is shown, using random coding arguments, that these codes have a large free MSED on the average.

A minimal systematic encoder can be implemented from (19), since $h^0_0 = 1$ [1]. The encoding equations are

$$z^j(D) = x^j(D), 1 \leq j \leq k \;, \tag{20a}$$

$$z^0(D) = H^{\tilde{k}}(D)x^{\tilde{k}}(D) \oplus \cdots \oplus H^1(D)x^1(D) \oplus (H^0(D) \oplus 1)z^0(D). \tag{20b}$$

An encoder implementation using (20) is shown in Figure 11.

For all codes with $\nu = 1$ and for some codes with $\nu > 1, \tilde{k} = \nu$. For these codes we cannot restrict h^j_0, for $1 \leq j \leq \tilde{k}$. This is because $\tilde{k}$ checked bits require at least $\tilde{k}$ terms in $H^j(D)$, for $1 \leq j \leq \tilde{k}$, that are variable. If there are not enough variables, then there will be some non-zero $x^{\tilde{k}} = [x^{\tilde{k}}, \ldots, x^2, x^1]$ such that $\sum_{j=1}^{\tilde{k}} H^j(D)x^j = 0 \pmod 2$. That is, there will be more than $2^{k-\tilde{k}}$ parallel transitions between states in the trellis. To avoid this problem, when $\tilde{k} = \nu$, we use (19) without any restrictons. In this case, the MSED between all possible non-parallel coded sequences is at least $\Delta^2_q + \Delta^2_{q+1}$, since the MSED between paths leaving a state is Δ^2_q (since $h^1_0 \in \{0,1\}$, for $1 \leq j \leq \tilde{k}$) and between paths entering a state is Δ^2_{q+1} (since $h^j_\nu = 0$, for $1 \leq j \leq \tilde{k}$).

signal set mapper takes the I bits for each 2-D signal point and produces the required real and imaginary (or amplitude and phase) components for a modulator.

Example 3.2

In this example, we describe how to implement a particular code. The code is used with a 3×8PSK signal set. Thus $L = 3$ and $I = 3$. We also choose $q = 1$, so that a 2.33 bit/T (rate 7/8) code is formed. The partition that is used is given in Table 3(b), from which we obtain $p_0 = 2, p_1 = 3$, and $p_2 = 6$. The code is 90° transparent, so that $d = 1$ and $s = 2$. Therefore $c_0 = p_1 - q = 2$, and $c_1 = p_2 - q = 5$. Thus bits w^2 and w^5 are precoded using a modulo-4 adder. Since $c_0 > 0$, the precoder given in Figure 10(a) is used. For this code, $\tilde{k} = 2$ and the parity check polynomials are $H^0(D) = D^4 \oplus D^2 \oplus D \oplus 1, H^1(D) = D$, and $H^2(D) = D^3 \oplus D^2$. Excluding the parallel to serial converter and the 2-D signal mapper, the encoder is shown in Figure 12. This code has 16 states ($\nu = 4$). Note that the multi-D signal set mapper does not correspond exactly to Figure 7(b), since $q = 1$.

3.4 Convolutional Encoder Effects on Transparency

The convolutional encoder can affect the total transparency of the system. The method used to determine transparency is to examine the parity check equation and the bits that are affected by a phase rotation. A code is transparent if its parity check equation, after substituting $z_j(D)$ with $z_r^j(D)$, for $0 \leq j \leq \tilde{k}$ (the rotated sequences), remains the same. There are normally at most I bits that are affected by a phase rotation, $z^{b_0}, \ldots, z^{b_{I-1}}$, $b_j = p_j - q$, for $0 \leq j \leq I - 1$. That is,

$$\begin{aligned} z_r^{b_0} &= z^{b_0} \oplus 1, & \text{(21a)} \\ z_r^{b_1} &= z^{b_1} \oplus z^{b_0}, & \text{(21b)} \\ z_r^{b_2} &= z^{b_2} \oplus z^{b_0} \cdot z^{b_1}, & \text{(21c)} \\ &\vdots \end{aligned}$$

Assume that the largest value of $b_j \leq \tilde{k}$ is b_0. This implies that only one term in the parity check equation is affected by a phase rotation. The other bits have no effect since they are not checked by the encoder, i.e., $b_j > \tilde{k}$ for $1 \leq j \leq I - 1$. The parity check equation after a phase rotation of Ψ then becomes

$$\begin{aligned} &H^{\tilde{k}}(D)z^{\tilde{k}}(D) \oplus \cdots \oplus H^{b_0}(D)[z^{b_0}(D) \oplus 1(D)] \oplus \cdots \oplus H^0(D)z^0(D) = 0(D), \\ &H^{\tilde{k}}(D)z^{\tilde{k}}(D) \oplus \cdots \oplus H^{b_0}(D)z^{b_0}(D) \oplus \cdots \oplus H^0(D)z^0(D) = E[H^{b_0}(D)](D), \quad (22) \end{aligned}$$

where $E[H^{b_0}(D)]$ is the modulo-2 number of non-zero terms in $H^{b_0}(D)$ and $1(D) = \sum_{j=-\infty}^{\infty} D^j$ is the all one sequence (i.e., $E[H^{b_0}(D)](D) = H^{b_0}(D)1(D)$). Thus if there is an even number of terms in $H^{b_0}(D)$, (22) is the same as (18). That is, the code is transparent to integer multiples of Ψ phase rotations of the signal set. However, if there is an odd number of terms in $H^{b_0}(D)$, then $E[H^{b_0}(D)] = 1$ and the coset of the convolutional code is produced. Even though the two equations are closely related, the codes are quite different and a decoder is not able to produce correctly decoded data from a Ψ phase rotation of the signal set.

of terms in $H^{b_0}(D)$, (22) is the same as (18). That is, the code is transparent to integer multiples of Ψ phase rotations of the signal set. However, if there is an odd number of terms in $H^{b_0}(D)$, then $E[H^{b_0}(D)] = 1$ and the coset of the convolutional code is produced. Even though the two equations are closely related, the codes are quite different and a decoder is not able to produce correctly decoded data from a Ψ phase rotation of the signal set.

Now assume that the first two terms are affected by a phase rotation, i.e., the largest value of $b_j \le \tilde{k}$ is b_1. The terms in the parity check polynomial $H^{b_0}(D)z^{b_0}(D) \oplus H^{b_1}(D)z^{b_1}(D)$ now become

$$[H^{b_0}(D) \oplus H^{b_1}(D)]z^{b_0}(D) \oplus H^{b_1}(D)z^{b_1}(D) \oplus E[H^{b_0}(D)](D).$$

In this case the parity check equation is different after a phase rotation (even if $E[H^{b_0}(D)] = 0$). This means that the code is not transparent to a Ψ phase rotation, but it could be transparent to 2Ψ or 4Ψ phase rotations. This is because the phase rotation equations reduce to

$$\begin{aligned} z_r^{b_0} &= z^{b_0}, \ldots, \ z_r^{b_{d-1}} = z^{b_{d-1}}, \\ z_r^{b_d} &= z^{b_d} \oplus 1, \ z_r^{b_{d+1}} = z^{b_{d+1}} \oplus z^{b_d}, \ldots \end{aligned}$$

for a $2^d\Psi$ phase rotation, where $d = 1$ or 2. If there is an even number of terms in $H^{b_1}(D)$, then $d = 1$. This is because an even number of terms in $H^{b_1}(D)$ cancels the effect on $z^{b_1}(D)$ when the signal set is rotated by 2Ψ. That is, the code is transparent to integer multiples of 2Ψ phase rotations, but not to multiples of Ψ. If there is an odd number of terms in $H^{b_1}(D)$, this cancellation effect does not occur, implying that $d = 2$ and the phase transparency is 4Ψ.

In general, if the largest value of $b_j \le \tilde{k}$ is b_f, then $d = f + E[H^{b_f}(D)]$. We can then determine those bits z^{c_j} which are affected by a $2^d\Psi$ phase rotation, i.e., $c_j = b_{j+d} = p_{j+d} - q$, for $0 \le j \le s-1$, where $s = I - d$.

Example 3.3

For the code given in Example 3.2, $\tilde{k} = 2, I = 3$, and $q = 1$. Thus $b_0 = 1, b_1 = 2$, and $b_2 = 5$. Since the largest value of $b_j \le 2$ is b_1, then $f = 1$. Therefore $d = 1 + E[H^{b_1}(D)] = 1 + E[D^3 \oplus D^2] = 1$. Thus the code is 90° transparent, and $c_0 = 2$ and $c_1 = 5$.

3.5 Systematic Search for Good Small Constraint Length Codes

An approximate lower bound for the symbol error probability [1] of a multi-D code is given by

$$P_s(e) \gtrsim \frac{N_{free}}{L} Q\left(\sqrt{\frac{d_{free}^2 R_{eff}}{2} \frac{E_b}{N_0}}\right), \tag{23}$$

where E_b/N_0 is the energy per information bit to single sided noise density ratio and $Q(\cdot)$ is the complementary error function. In (23), the division by L normalizes the average number of errors per multi-D signal to that of a 2-D signal set.

For each multi-D signal set considered, there are a number of code rates which can be achieved. As ν is increased, a comprehensive code search becomes time consuming due to the greater complexity of each code. We have thus limited our search to $\nu + \tilde{k} \leq 10$. (The number of checked bits $\tilde{k}$ also affects the complexity of the code search.) As indicated by (23), the criteria used to find the best codes are the free MSED (d_{free}^2) and the number of nearest neighbors (N_{free}). We have also included the code transparency (d) as a criteria in the code serach. The code search algorithm that was implemented is similar to that in [1], but with a number of differences which include the extra criteria mentioned above.

The actual code search involves using a rate $\tilde{k}/(\tilde{k}+1)$ code. Thus two separate notations are used to distinguish the rate $k/(k+1)$ encoder and the simplified rate $\tilde{k}/(\tilde{k}+1)$ encoder. For the rate $k/(k+1)$ encoder, we have $\mathbf{x}_n = [x_n^k, \ldots, x_n^1]$ (the input to the encoder) and $\mathbf{z}_n = [z_n^k, \ldots, z_n^1, z_n^0]$ (the mapped bits or encoder output) at time n. Also, $\mathbf{e}_n = [e_n^k, \ldots, e_n^1, e_n^0]$ is the modulo-2 difference between two encoder outputs $\mathbf{z}_n$ and $\mathbf{z}'_n$ at time n, i.e., $\mathbf{e}_n = \mathbf{z}_n \oplus \mathbf{z}'_n$. Note that there are 2^{k+1} combinations of $\mathbf{z}_n$ and $\mathbf{z}'_n$ that give the same $\mathbf{e}_n$. For the rate $\tilde{k}/(\tilde{k}+1)$ code, we denote reduced versions of $\mathbf{x}_n, \mathbf{z}_n$, and $\mathbf{e}_n$ as $\tilde{\mathbf{x}}_n = [x_n^{\tilde{k}}, \ldots, x_n^1], \tilde{\mathbf{z}}_n = [z_n^{\tilde{k}}, \ldots, z_n^1, z_n^0]$, and $\tilde{\mathbf{e}}_n = [e_n^{\tilde{k}}, \ldots, e_n^1, e_n^0]$, respectively.

In order to find d_{free}^2 for a particular code, the squared Euclidean weights (SEW) $w^2(\mathbf{e}_n)$ are used. As defined in [1], $w^2(\mathbf{e}_n)$ is the MSED between all combinations of $a(\mathbf{z}_n)$ and $a(\mathbf{z}'_n)$ such that $\mathbf{e}_n = \mathbf{z}_n \oplus \mathbf{z}'_n$ and $a(\mathbf{z}_n)$ is the actual L×MPSK signal point. This can be defined as

$$w^2(\mathbf{e}_n) = \min_{all\ \mathbf{z}_n} d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)], \tag{24}$$

where $d^2[a(\mathbf{z}_n), a(\mathbf{z}'_n)]$ is the SED between $a(\mathbf{z}_n)$ and $a(\mathbf{z}'_n)$. One can then use the all zero path as a reference to find d_{free}^2 in a code search, i.e.,

$$d_{free}^2 = \min \sum_n w^2(\mathbf{e}_n), \tag{25}$$

where the minimization is over all allowable code sequences with the exception of the all-zero sequence. We can use (25) to find d_{free}^2 provided that the minimization of (24) does not depend on z_n^0, as shown by Ungerboeck [1].

Although the minimization of (24) does not depend on z_n^0 for 1×MPSK signal sets, it cannot be assumed that this also applies to L×MPSK for $L \geq 2$. By expressing $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$ directly in terms of $\mathbf{z}_n$ and $\mathbf{e}_n$, it can be shown that 3×4PSK (I), 3×8PSK (I and II), and 3×16PSK (I, II, and III) all depend on z_n^0. This implies that (25) becomes a lower bound in these cases. However, due to the large number of parallel transitions for these codes, we can still determine d_{free}^2 (and N_{free}) using a slightly modified version of (25).

Since there are 2^{k+1} values of $\mathbf{e}_n$, there are a total of 2^{2k+2} computations required to find all the values of $w^2(\mathbf{e}_n)$. For example, a rate 11/12 code with 4×8PSK modulation requires nearly 17 million computations. This can be reduced by letting $z_n^0 = 0$ (or 1) and minimizing (24) over all $\mathbf{z}_n = [z_n^k, \ldots, z_n^1, 0]$. This reduces the number of computations to 2^{2k+1}. In fact, it is possible to even further decrease the number of computations. Using some difficult algebraic manipulations, it can be shown that the L output bits z_n^p corresponding to cosets t^p with some components equal to 2^{I-1} can all be set to zero. For example, the 4×8PSK signal set with $q = 0$ can have bits z_n^7, z_n^9, z_n^{10}, and z_n^{11} all set to 0 when minimizing (24).

This is due in part to the MPSK signals being antipodal for these values. Thus the total number of computations can be reduced to 2^{2k-L+1}.

In order to reduce the time needed to find d^2_{free}, we note that the trellis is equivalent to a rate $\tilde{k}/(\tilde{k}+1)$ code with $2^{k-\tilde{k}}$ parallel transitions. Also, there are $2^{\tilde{k}+1}$ different sets of parallel transitions. If the minimum SEW is found for each of these sets of parallel transitions, the code search is greatly simplified, since the search for a rate $\tilde{k}/(\tilde{k}+1)$ code is all that is needed and $\tilde{k}$ is usually small. Thus, the SEW's required for a rate $\tilde{k}/(\tilde{k}+1)$ code search are

$$w^2(\tilde{\mathbf{e}}_n) = \min w^2(\mathbf{e}_n), \tag{26}$$

where the minimization is over all $[e_n^k, \ldots, e_n^{\tilde{k}+1}]$. We define the free MSED of this rate $\tilde{k}/(\tilde{k}+1)$ code as

$$\tilde{d}^2_{free} = \min \sum_n w^2(\tilde{\mathbf{e}}_n), \tag{27}$$

where the minimization is over all allowable code sequences $(\tilde{\mathbf{e}}(D))$ defined by

$$\tilde{\mathbf{e}}(D) = \tilde{\mathbf{e}}_1 D \oplus \tilde{\mathbf{e}}_2 D^2 \oplus, \cdots \oplus \tilde{\mathbf{e}}_N D^N,$$

for $\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_N \neq \mathbf{0}$, and $N \geq 2$. The code sequences of length $N = 1$ are the parallel transitions, where the MSED is the MSSD of the parallel transitions. A code might have $\tilde{d}^2_{free}$ larger than the MSSD of the parallel transitions, implying that d^2_{free} occurs along the parallel transitions. With $\tilde{k}$ checked bits and a rate $\tilde{k}/(\tilde{k}+1)$ code, the MSSD of the parallel transitions is $\Delta^2_{q+\tilde{k}+1}$. Thus we can express d^2_{free} as

$$d^2_{free} = \min(\tilde{d}^2_{free}, \Delta^2_{q+\tilde{k}+1}). \tag{28}$$

The best value of $\tilde{k}$ can be determined from the free MSED of the best code for the previous value of ν. The search starts with $\nu = 1$ and $\tilde{k} = 1$, and we find the code with the best d^2_{free} and N_{free}. We then increase ν by one and determine $\tilde{k}$ as follows. If d^2_{free} for the previous best code was $\tilde{d}^2_{free}$, then $\tilde{k}$ remains the same. This is because the limit of the parallel transitions $(\Delta^2_{q+\tilde{k}+1})$ has not yet been reached and the trellis connectivity needs to be reduced in order to increase d^2_{free} or reduce N_{free}. If d^2_{free} for the previous best code was $\Delta^2_{q+\tilde{k}+1}$, then $\tilde{k}$ is increased by one from the previous value; otherwise, d^2_{free} and N_{free} would remain the same. If $\tilde{d}^2_{free} = \Delta^2_{q+\tilde{k}+1}$ for the previous best code, then $\tilde{k}$ can remain the same or increase by one. Both values of $\tilde{k}$ should be tried in order to find the best code. The best code is then found for this value of ν and $\tilde{k}$, and the above process is repeated for each increasing value of ν.

As can be seen from (24), there may be some values of $\mathbf{e}_n$ and $\mathbf{z}_n$ for which $w^2(\mathbf{e}_n) < d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$. The *number of nearest neighbors* for $\mathbf{e}_n$ (denoted $m(\mathbf{e}_n)$) is defined as the average number of times that $w^2(\mathbf{e}_n)$ equals $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$. If $w^2(\mathbf{e}_n)$ equals $d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]$ for all values of $\mathbf{z}_n$, then $m(\mathbf{e}_n) = 1$. For example, in naturally mapped 8PSK it is found that for $\mathbf{e}_n = [0\ 1\ 1]$ and $[1\ 1\ 1], d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)] = 0.586$ for four values of $\mathbf{z}_n$ and 3.414 for the other four values of $\mathbf{z}_n$. Thus $m(\mathbf{e}_n) = 0.5$ for $\mathbf{e}_n = [0\ 1\ 1]$ and $[1\ 1\ 1]$. For all other values of $\mathbf{e}_n$, it can be shown that $m(\mathbf{e}_n) = 1$. Zehavi and Wolf

[21] give a general approach to determining the full code distance spectrum, whereas we are only interested in the number of nearest neighbors.

We can state this generally as follows. Let the number of bits in $\mathbf{z}_n$ that are varied to find $w^2(\mathbf{e}_n)$ be b. Then

$$m(\mathbf{e}_n) = \sum u\left(w^2(\mathbf{e}_n) - d^2[a(\mathbf{z}_n), a(\mathbf{z}_n \oplus \mathbf{e}_n)]\right) 2^{-b}, \tag{29}$$

where $u(\cdot)$ is the unit step function and the summation is over all the bits in $\mathbf{z}_n$ that are varied to find $w^2(\mathbf{e}_n)$. Normally $b = k+1$, but this can be reduced to $b = k - L$ for the reasons mentioned previously.

For the simplified rate $\tilde{k}/(\tilde{k}+1)$ code, $m(\tilde{\mathbf{e}}_n)$ is the sum of all the $m(\mathbf{e}_n)$'s for which $w^2(\tilde{\mathbf{e}}_n) = w^2(\mathbf{e}_n)$, i.e.,

$$m(\tilde{\mathbf{e}}_n) = \sum u\left(w^2(\tilde{\mathbf{e}}_n) - w^2(\mathbf{e}_n)\right) m(\mathbf{e}_n), \tag{30}$$

where the summation is over all $[e_n^k, \ldots, e_n^{\tilde{k}+1}]$. We can think of $m(\tilde{\mathbf{e}}_n)$ as the total average number of nearest neighbors along each set of parallel transitions.

The number of nearest neighbors for the MSSD $\Delta^2_{q+\tilde{k}+1}$ is

$$N_\Delta = \sum u\left(\Delta^2_{q+\tilde{k}+1} - w^2(\mathbf{e}_n)\right) m(\mathbf{e}_n), \tag{31}$$

where the summation is over all $\mathbf{e}_n = [e_n^k, \ldots, e_n^{\tilde{k}+1}, 0, \ldots, 0]$. The number of nearest neighbors for paths with SED $\tilde{d}^2_{free}$ can be calculated using $m(\tilde{\mathbf{e}}_n)$ as follows:

$$\tilde{N}_{free} = \sum_{\alpha=1}^{A} \prod_{n=1}^{N_\alpha} m(\tilde{\mathbf{e}}_n), \tag{32}$$

where N_α is the length of a path α that has a SED of $\tilde{d}^2_{free}$ and A is the number of paths that have a SED of $\tilde{d}^2_{free}$. If d^2_{free} occurs along the parallel transitions, $N_{free} = N_\Delta$, and we define the next nearest free SED and number of nearest neighbors as $d^2_{next} = \tilde{d}^2_{free}$ and $N_{next} = \tilde{N}_{free}$, respectively. (Note that d^2_{next} and N_{next} may not be the true next nearest paths, since there may be some closer paths occuring along the parallel transitions.) When there are several codes that have the same free MSED and number of nearest neighbors, the "next nearest" values are used in code selection. When d^2_{free} occurs along paths with SED $\tilde{d}^2_{free}, N_{free} = \tilde{N}_{free}$. The next nearest values in this case are not given in the code tables. If $\tilde{d}^2_{free} = \Delta^2_{q+\tilde{k}+1}$, then $N_{free} = N_\Delta + \tilde{N}_{free}$.

<u>Example 3.4</u>

In Example 3.2 we have a $\tilde{k} = 2, q = 1$, rate 7/8 (2.33 bit/T) code with a 3×8PSK (II) signal set. After determining the mapping of the signal set, (24) was used to find the SEW's for each signal point. Equation (26) determines the $w^2(\tilde{\mathbf{e}}_n)$'s that were used to find the best rate 2/3 codes. For these codes $d^2_{free} = \Delta^2_{q+\tilde{k}+1} = \Delta^2_4 = 4.0$. Using (31) we determined that N_{free} is 15 (after normalizing, there are only 5 paths per 2-D symbol). In the code search for the best rate 2/3 codes, there were many codes which had $d^2_{next} = \tilde{d}^2_{free} = 4.343$. Thus (32) was used to determine N_{next} for each best code. Table 10 gives the values of $w^2(\tilde{\mathbf{e}}_n)$ and

$m(\tilde{\mathbf{e}}_n)$ for each $\tilde{\mathbf{e}}_n$ that were used in the code search. The best code with a transparency of 90° was found to have $N_{next} = 24$.

In order to reduce the number of codes that must be tested in our code search algorithm, rejection rules were used. As in Rule 1 of [1], time reversal of the parity check polynomials was used to reject codes. Even though $w^2(\tilde{\mathbf{e}}_n)$ and $m(\tilde{\mathbf{e}}_n)$ are used to find the best codes, Rule 2 in [1] can still be exploited, provided that $w^2(\tilde{\mathbf{e}}_n) = \Delta^2_{r(\tilde{\mathbf{e}}_n)+q'}$, where $r(\tilde{\mathbf{e}}_n)$ is the number of trailing zero's in $\tilde{\mathbf{e}}_n$. When this is not true, it may still be possible to find some combinations of the parity check polynomials that can be rejected (this was also implemented in our code search). Rule 3 in [1] was also used to eliminate codes.

In the code search, a rate $\tilde{k}/(\tilde{k}+1)$ code is searched for a particular ν. Before finding $\tilde{d}^2_{free}$, the code search program checks to make sure that the code only produces sequences with length $N \geq 2$. If for some input $\tilde{x}_n \neq \mathbf{0}$, the inputs to the systematic encoder are all zero, the state of the encoder goes from one state to the next as if a zero input had occurred. Thus parallel transitions will occur in the rate $\tilde{k}/(\tilde{k}+1)$ code, which should not have parallel transitions. Therefore, in the code search, codes at level i $(1 \leq i \leq \tilde{k})$ were rejected if for some $[x^i, \ldots, x^1] \neq \mathbf{0}$, $\sum_{j=1}^{i} x^j H^j(D) \pmod 2 = 0(D)$.

Two programs were used in the code search, one for codes with $\nu > \tilde{k}$ and the other for codes with $\nu = \tilde{k}$. For specific values of I, L, and $q, y^q(z)$, for $0 \leq z \leq 2^{IL-q} - 1$, was generated using the coset representatives t^p, for $0 \leq p \leq IL - 1$, that are given in Tables 5, 6, and 7. The squared Euclidean weights $w^2(\mathbf{e}_n)$ were then calculated using (24) for all $\mathbf{e}_n$. Since the value of $\tilde{k}$ can change with each ν, $w^2(\tilde{\mathbf{e}}_n)$ and $m(\tilde{\mathbf{e}}_n)$ were computed, if necessary, as the program went from the smallest to the largest ν.

The code search used the various rejection rules before the time consuming tasks of finding $\tilde{d}^2_{free}$ (using the bi-directional search algorithm [22]) and N_{free} (using a technique based on the Viterbi algorithm). The rejection rules were organized so that the best codes for each of the two possible phase transparancies were found. The code search found those codes which had the largest free distance (for a particular transparency). If a code was found to have its free MSED equal to or greater than the previous best code, $\tilde{N}_{free}$ was determined and this code was listed if either its $\tilde{d}^2_{free}$ or $\tilde{N}_{free}$ had improved over the previous best code.

The octal code generators were then listed along with their $\tilde{d}^2_{free}$, $\tilde{N}_{free}$, and phase transparency d. A small list of codes was produced (for each code search) from which the best codes could be chosen. Every time that $\tilde{k}$ is increased by one in the code search (which is done automatically), the program determines and lists $\Delta^2_{q+\tilde{k}+1}$ and N_Δ for use in the code tables.

The asymptotic coding gain γ of each code compared to the uncoded case, as shown in the code tables, is

$$\gamma = 10\log_{10}(d^2_{free}/d^2_u)\ (dB), \tag{33}$$

where d^2_u is the smallest MSSD of an equivalent uncoded 2-D or multi-D scheme. In nearly all cases, $d^2_u = \Delta^2_{q+1}$. For codes with a non-integer R_{eff}, no equivalent 1×MPSK scheme exists which has the same R_{eff}, and so the equivalent uncoded multi-D signal set is used instead. For the 4×8PSK signal set with $q = 3, R_{eff} = 2$ bit/T. Thus, a natural comparison would be against uncoded 4PSK, which has $d^2_u = 2$. (In this case, $\Delta^2_{q+1} = 2.343$, which is inconsistent with other codes that also have $R_{eff} = 2$ bit/T.) The asymptotic coding gains compared to uncoded (M/2)-PSK are found by adding to γ the appropriate correction factor

$$\gamma_{M/2} = 10\log_{10}\left(\frac{R_{eff}}{(I-1)}\frac{d_u^2}{\delta_1^2}\right) \ (dB), \tag{34}$$

as shown in the code tables. The transparency (in degrees) is also given for each code. The parity check polynomials are expressed in octal notation in the code tables, e.g., $H^0(D) = D^6 + D^4 + D^2 + D + 1 \equiv (001\ 010\ 111)_2 \equiv (127)_8$.

In Tables 11, 15, and 19, codes for TC-1×4PSK (rate 1/2 4PSK), TC-1×8PSK (rate 2/3 8PSK), and TC-1×16PSK (rate 3/4 16PSK), respectively, are presented. These tables give the best code for each phase transparency, which (to the best of our knowledge) have not been previously published. The best codes, without regard for phase transparency, were originally published by Odenwalder [15] for 4PSK (with the codes in non-systematic form), by Ungerboeck [1,4] for 8PSK, and by Wilson, et. al. [6] for 16PSK.

Tables 12, 16, and 20 list the TC-2×4PSK codes (rates of 1.5 and 1.0 bit/T), the TC-2×8PSK codes (2.5 and 2.0 bit/T), and the TC-2×16PSK codes (3.5 and 3.0 bit/T), respectively. Tables 13, 17, and 21 list the TC-3×4PSK codes (1.67, 1.33, and 1.0 bit/T), the TC-3×8PSK codes (2.67, 2.33, and 2.0 bit/T), and the TC-3×16PSK codes (3.67, 3.33, and 3.0 bit/T), respectively. Tables 14, 18, and 22 list the TC-4×4PSK codes (1.75, 1.5, 1.25, and 1.0 bit/T), the TC-4×8PSK codes (2.75, 2.5, 2.25, and 2.0 bit/T), and the TC-4×16PSK codes (3.75, 3.5, 3.25, and 3.0 bit/T), respectively.

Equivalent $R = 5/6$, TC-2×8PSK (2.5 bit/T) codes with up to 16 states have been found independently by Lafanechère and Costello [8] and by Wilson [9], although with reduced phase transparency. The 2 state TC-L×8PSK and TC-L×16PSK codes were also found by Divsalar and Simon [23].

In the code tables it can be seen that for the same complexity, there are usually two codes (and in some cases three codes) that are given. Note that the code with the worst phase transparency has a better free distance or a fewer number of nearest or next nearest neighbors. Thus, if phase transparency is not required, one should choose the less phase transparent code in order to obtain the maximum performance for a given complexity.

3.6 Decoder Implementation

When the Viterbi algorithm is used as the decoder, a measure of decoding complexity is given by $2^{\nu+\tilde{k}}/L$. This is the number of distinct transitions in the trellis diagram for any TCM scheme normalized to a 2-D signal set. The maximum bit rate of the decoder is kf_d, where f_d is the symbol speed of the decoder. Since k is quite large for multi-D signal sets (at least $(I-1)L$), high bit rates can be achieved. For example, a Viterbi decoder has been constructed for a rate 7/9 periodically time varying trellis code (PTVTC) with $\nu = 4, \tilde{k} = 2$, and 8PSK modulation [24]. This decoder has $f_d = 60$ MHz and a bit rate of 140 Mbit/s. However, with the equivalent rate 7/8 code with 3×8PSK modulation, the bit rate will be $L = 3$ times as fast, i.e., 420 Mbit/s. The branch metric calculator, though, will be more complicated due to the larger number of parallel transitions between states. Alternatively, one could build a decoder operating at a 20 MHz speed and achieve the same bit rate of 140 Mbit/s. In addition to providing decreased decoder complexity, this multi-D code has an asymptotic coding gain which is 0.56 dB greater and is 90 transparent, compared with a 180 transparency for the PTVTC [25].

Although the decoding complexity of the Viterbi algorithm is measured in terms of $2^{\nu+\tilde{k}}/L$, for multi-D schemes the complexity of subset (parallel transition) decoding must also be taken into account due to the large number of parallel transitions.

The Viterbi decoder must find which of the $2^{k-\tilde{k}}$ parallel transitions is closest, in a maximum likelihood sense, to the received signal. A brute force method would be to determine the metric for each of the $2^{k-\tilde{k}}$ paths and then find the minimum. This would involve at least $2^{k-\tilde{k}}-1$ comparisons. Since there are $2^{\tilde{k}+1}$ sets of parallel transitions, a total of $2^{k+1}-2^{\tilde{k}+1}$ comparisons would be required. For large k and small $\tilde{k}$, this is an unacceptably large number of computations.

Fortunately, as shown in [13] for binary lattices, it is possible to greatly reduce the number of computations required. In fact, the decoding scheme becomes very similar to Viterbi decoding, except that finite length sequences are used.

To illustrate this we will present the decoding scheme for TC-2×8PSK parallel transitions with $\tilde{k}=2$ and an efficiency of 2.5 bit/T (a rate 5/6 code). There are eight sets of parallel transitions, with eight paths in each set. Figure 13 shows the parallel transition decoding trellis for $\tilde{\mathbf{z}}=[000]$ (i.e., the three lsb's are set to zero). In Figure 1, we use the notation $A0$ to indicate the whole 8PSK signal set, which divides into $B0$ and $B1$ (4PSK signal sets rotated 45° from each other). $B0$ divides into $C0$ and $C2$ (2PSK signal sets rotated 90° from each other), and $B1$ divides into $C1$ and $C3$. This notation is also used in [1] for partitioning an 8PSK signal set. Each segment in Figure 13 thus represents two parallel lines. The length of this trellis equals the dimensionality $L=2$ of the signal set.

The path $C0\times C0$ corresponds to those four paths that have $z^3=0$ and $C2\times C2$ corresponds to those four paths that have $z^3=1$, giving a total of eight paths. To decode, hard decisions can be made for $C0$ and $C2$ for each time period, from which the values of z^4 and z^5 can be determined. For example, say that $C0\times C0$ decodes into the points 04, with a metric of m_0, and $C2\times C2$ decodes into the points 66, with a metric of m_1, where the metrics are the sum of the Euclidean distances (or log-likelihood metrics for a quantized channel) from the first and second received points. After comparing the two metrics, if $m_0<m_1$, then $z^3=0$ and the point 04 would give $z^4=1$ and $z^5=0$ (see Table 1). If $m_0>m_1$, then $z^3=1$, and the point 66 would give $z^4=0$ and $z^5=1$. This is equivalent to the add-compare-select (ACS) operation within a Viterbi decoder.

To decode the other sets of parallel transitions, the cosets formed by z^0, z^1, and z^2 can be added to the trellis paths $C0\times C0$ and $C2\times C2$ to form the required trellis. This is illustrated in Figure 14, where the ending state in the trellis indicates which set of parallel transitions is being decoded. In this example, there are a total of eight hard comparisons and eight ACS type comparisons. These 16 comparisons compare with the 56 comparisons required in a brute force approach, a 3.5 times reduction.

The above maximum likelihood method can be applied to other codes where a Viterbi like decoder can be used to decode the parallel transitions. With this method, the complexity of decoding the parallel transitions can approach the complexity of the rate $\tilde{k}/(\tilde{k}+1)$ Viterbi decoder. A simpler approach may be with large look-up tables using ROM's. The ROM itself would output the $k-\tilde{k}$ bits of the chosen path, along with the branch metric for that path. For the TC-2×PSK example given previously, we could use one ROM for each set of parallel transitions. If the ROM's had eight bit words, then three bits could be used for the decision, and the remaining five bits for the branch metric. A total of eight ROM's would then be required, one for each set of the parallel transitions.

When using ROM's, it is desirable to reduce the number of bits (b) required to represent each received 2-D signal point, since there are a total of bL bits required to address the ROM. One way to reduce b is to convert the 'checkerboard' (rectangular) type decision boundaries that result from separate quantization of the inphase (I) and quadrature (Q) components to 'dartboard' (radial) type decision boundaries. For example, if four bits are used in I and Q for an 8PSK signal with checkerboard decision boundaries, a dartboard pattern as shown in Figure 15 may be used instead with a total of five bits to represent each point (a reduction of three bits). A ROM may be used to do the conversion, or the dartboard pattern may be already available as polar coordinates from a digital demodulator.

A problem with TC-L×MPSK is the need to synchronize the decoder with the L 2-D symbols on each trellis branch. For $q=0$, most codes are fully transparent. The decoder performance can then be used to find the correct synchronization with the received sequence. For $q>0$, many codes are not fully transparent, and the decoder will need to synchronize to one of the 2^dL possibilities (which can be quite large for some codes). However, one can take advantage of the fact that not all signal points are used for $q>0$. For example, the 2×8PSK signal set with $q=1$ consists of the signal sets B0×B0 or B1×B1. The synchronizer would find the smallest distance between a received pair of points and the expected signal set. These distances would then be accumulated over a sufficient length of time to make a reliable decision on the symbol timing.

If we let each signal point be represented by its phase (since the amplitude is constant for 8PSK), we can write $B0=\{0°,90°,180°,270°\}$, and $B1=\{45°,135°,225°,315°\}$. Let ϕ_n^1 and ϕ_n^2 represent the phase of the first and second received symbols, respectively. The synchronizer distance metric is then given by

$$\Phi_n = \min_{i\in\{0,1\}} \left(\min_{\alpha\in Bi} |\phi_n^1 - \alpha| + \min_{\beta\in Bi} |\phi_n^2 - \beta| \right).$$

In the synchronized noiseless case, Φ_n will equal zero. In the non-synchronized noiseless case, there are two possible outcomes for Φ_n, i.e., complete matchup ($\Phi_n = 0°$) and only one signal is matched ($\Phi_n = 45°$). If each possiblity is equally likely, then the average value of Φ_n is 22.5° . With noise, Φ_n can be accumulated over a sufficient length of symbols to take advantage of this average phase distance between the non-synchronized and synchronized cases to reliably determine symbol synchronization. This symbol synchronization is independent of the Viterbi decoder, so the decoder must only determine phase synchronization.

3.7 Discussion

In order to make a comparison of all the codes listed, a plot of *nominal coding gain* $\gamma^* = 10\log_{10} d^2_{free}$ verses complexity ($\beta = \log_2(2^{\nu+\tilde{k}}/L) = \nu + \tilde{k} - \log_2 L$) for each code found is made. These plots are given in Figure 16 for effective rates of 1.0 (with 4PSK modulation), 2.0 (8PSK), and 3.0 bit/T (16PSK), Figure 17 for effective rates of 1.5 (4PSK), 2.5 (8PSK), and 3.5 bit/T (16PSK), and Figure 18 (for the remaining rates). (Note that these graphs do not take into account the additional complexity due to parallel transitions.) Some one state ('uncoded') codes are included as well. These one state codes correspond to block coded (or multilevel) schemes that have recently become an active research area [26-30]. Although the multi-D one state codes have negative complexity (compared to trellis codes), they can achieve coding gains above 0 dB.

Note from Figure 16 for TC-L×8PSK, $R_{eff} = 2.0$ bit/T, and $\nu = 1$, that as L increases the complexity decreases and γ^* increases, eventually reaching 6.0 dB for $L = 4$. Thus, for the 8-D signal set, the complexity factor can be reduced by a factor of four, while maintaining ν^*, compared to the TC-1×8PSK code with $\nu = 2$. Beyond $\beta = 4$ (and $\gamma^* = 6.0$ dB), increases in asymptotic coding gain are achieved with the new codes that have been found. With $L = 4$, a ceiling of $\gamma^* = 9.0$ dB will be reached due to the nature of the set partitioning. It would seem that very complex codes are required ($\beta > 15$) if this 9.0 dB limit is to be exceeded.

Figure 16 also shows the L×16PSK codes with effective rates of 3.0 bit/T. For small β, the same effect observed for TC-L×8PSK and 2.0 bit/T occurs. That is, β decreases and γ^* increases as L increases. Between $\beta = 3$ and $\beta = 9$, the $L = 1$ and $L = 2$ codes are very close.

Figure 18 illustrates the wide range of performance that can be achieved with the codes found. One can choose from a high rate code with 3.75 bit/T (but requiring a large amount of power) to a low rate code with 1.25 bit/T. In choosing a code, a designer may start with a required R_{eff} in order to obtain a certain bit rate through a bandwidth constrained channel. A trade-off can then be made between decoder complexity and the reduction in SNR that can be achieved with the codes found. Simulations or theoretical calculations of a few selected codes may also be made in order to obtain a more realistic assesment of the performance available.

Note that many codes have the same asymptotic coding gain for increasing complexity. In reality, these codes do increase in performance with increasing complexity due to a decrease in number of nearest neighbors. This is especially noticeable for low SNR where the effect of nearest neighbors becomes more important.

4 Conclusions

An efficient method of partitioning multi-dimensional MPSK signal sets has been presented that leads to easily implemeted multi-D signal set mappers. When these signal sets are combined with trellis codes to form a rate $k/(k+1)$ code, significant asymptotic coding gains in comparison to an uncoded system are achieved. These codes provide a number of advantages compared to trellis codes with 2-D signal sets. Most importantly, R_{eff} can vary from $I - 1$ to $I - (1/L)$ bit/T, allowing the coding system designer a greater choice of data rates without sacrificing data quality. As R_{eff} approaches I, though, increased coding effort (in terms of decoder complexity) or higher SNR is required to achieve the same data performance.

The analytical description of multi-D signal sets in terms of block code cosets, and the use of systematic convolutional encoding, has resulted in an encoder design (from the differential encoder to the 2-D signal set mapper) that allows many good codes to be found. This approach has also led to the construction of signal sets that allow codes to be transparent to multiples of $360°/M$ phase rotations. In general, increasing phase transparency usually results in lower code performance, due to more nearest or next nearest neighbors or smaller free distance.

Another advantage is decoder complexity. As a Viterbi decoder decodes k bits in each recursion of the algorithm, the large values of k of codes using multi-D signal sets allows very high bit rates to be achieved (compared to convolutional codes that map only into a 2-D signal set). The large number of branch metric computations can be reduced either through the use of a modified Viterbi algorithm or large look up tables. A method has been presented that uses the redundancy in some signal sets to achieve symbol synchronization at the decoder for codes that are not fully transparent.

Rate $k/(k+1)$ TC-L$\times$MPSK codes also have the advantage of being useful as inner codes in a high rate concatenated coding system with Reed-Solomon (RS) outer codes over $GF(2^k)$. If the inner decoder makes errors, one trellis branch error will exactly match one symbol in the outer RS code word. It is shown in [14] that the symbol oriented nature of TC-L$\times$MPSK inner codes can provide an improvement of up to 1 dB in the overall performance of a concatenated coding system when these codes replace bit oriented TC-1$\times$MPSK inner codes of the same rate.

Appendix

Differential Encoding and Decoding

Let the bit streams that are differentially encoded be $w^{c_0}(D), w^{c_1}(D), \ldots, w^{c_{s-1}}(D)$. We first assume that $c_0 > 0$ (i.e., the convolutional encoder output $z^0(D)$ is not affected by a phase rotation of $2^d\Psi$, where $d = I - s$). Let

$$w(D) = \sum_{i=0}^{s-1} 2^i w^{c_i}(D). \tag{A.1}$$

The differential encoder (or precoder) outputs are the bit streams $x^{c_0}(D), x^{c_1}(D), \ldots,$ $x^{c_{s-1}}(D)$ which go into the convolutional encoder. Similar to (A.1), we let

$$x(D) = \sum_{i=0}^{s-1} 2^i x^{c_i}(D). \tag{A.2}$$

For the noiseless channel, we let the Viterbi decoder output which goes into the differential decoder (or postcoder) be $x_r(D)$, and the output from the postcoder be $w_r(D)$. After a $2^d\Psi$ phase rotation, we have from Section 3.2 that

$$x_r(D) = x(D) + 1(D)(\text{ mod } S), \tag{A.3}$$

where $S = 2^s$ and $1(D)$ is the all ones sequence. For the postcoder, we desire that $w_r(D) = w(D)$ for all multiples of $2^d\Psi$ phase rotations. This is achieved by defining the postcoder equation as

$$w_r(D) = ((S-1)D + 1)x_r(D)(\text{ mod } S). \tag{A.4}$$

Substituting (A.3) into (A.4) we obtain

$$\begin{aligned} w_r(D) &= ((S-1)D+1)(x(D)+1(D)) && (\text{mod S}) \\ &= ((S-1)D+1)x(D) + ((S-1)D+1)1(D) && (\text{mod S}) \\ &= w(D) + (S-1)1(D) + 1(D) && (\text{mod S}) \\ &= w(D) + (S)1(D) && (\text{mod S}) \\ &= w(D), \end{aligned}$$

as required. Notice that since 1(D) is defined to be 1 for all time, then $D^i 1(D) = 1(D)$ for all i. In practical situations, the sequence added to $x(D)$ to form $x_r(D)$ is not constant, and will change with time (e.g., random phase slips within a demodulator). This will introduce short error bursts in $w_r(D)$ whenever a phase slip occurs due to the combined effect of decoding and postcoding. The precoder equation can be derived from (A.4) as

$$x(D) = Dx(D) + w(D)(\text{ mod } S). \tag{A.5}$$

We shall now consider the case when $c_0 = 0$, i.e., $z^0(D)$ is affected by a $2^d\Psi$ phase rotation. In this case we redefine $w(D)$ to be

$$w(D) = \sum_{i=1}^{s-1} 2^{i-1} w^{c_i}(D), \tag{A.6}$$

and $x(D)$ to be

$$x(D) = \sum_{i=1}^{s-1} 2^{i-1} x^{c_i}(D). \tag{A.7}$$

For this case, we have $2x_r(D) + z_r^0(D) = 2x(D) + z^0(D) + 1(D)$, where $x_r(D)$ and $z_r^0(D)$ are the inputs to the postcoder for a noiseless channel. Thus, similar to (A.4), the postcoder equation is defined to be

$$2w_r(D) = ((S-1)D + 1)(2x_r(D) + z_r^0(D)) \qquad (\bmod S). \tag{A.8}$$

Rearranging (A.8), we obtain the precoder equation

$$2x(D) = 2Dx(D) + 2w(D) + (D + S - 1)z^0(D) \qquad (\bmod S). \tag{A.9}$$

References

[1] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, Vol. IT-28, pp. 55-67, January 1982.

[2] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE J. Selected Areas in Commun.*, Vol. SAC-2, pp. 632-647, Sept. 1984.

[3] A. R. Calderbank and J. E. Mazo, "A new description of trellis codes," *IEEE Trans. Inform. Theory*, Vol. IT-30, pp. 784-791, Nov. 1984.

[4] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets: Parts I and II," *IEEE Commun. Magazine*, Vol. 25, pp. 5-21, Feb. 1987.

[5] D. P. Taylor and H. C. Chan, "A simulation study of two bandwidth efficient modulation techniques," *IEEE Trans. Commun.*, Vol. COM-29, pp. 267-275, March 1981.

[6] S. G. Wilson, H. A. Sleeper, P. J. Schottler, and M. T. Lyons, "Rate 3/4 convolutional coding of 16PSK: code design and performance study," *IEEE Trans. Commun.*, Vol. COM-32, pp. 1308-1315, Dec. 1984.

[7] A. R. Calderbank and N. J. A. Sloane, "Four-dimensional modulation with an eight state trellis code," *AT&T Tech. Journal*, Vol. 64, No. 5, pp. 1005-1018, May-June 1985.

[8] A. Lafanechère and D. J. Costello, Jr., "Multidimensional coded PSK systems using unit-memory trellis codes," *Proc. Allerton Conf. on Commun., Cont., and Comput.*, pp. 428-429, Monticello, IL, Sept. 1985.

[9] S. G. Wilson, "Rate 5/6 trellis-coded 8PSK," *IEEE Trans. Commun.*, Vol. COM-34, pp. 1045-1049, Oct. 1986.

[10] A. R. Calderbank and N. J. A. Sloane, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, Vol. IT-33, pp. 177-195, March 1987.

[11] L. F. Wei, "Trellis-coded modulation with multi-dimensional constellations," *IEEE Trans. Inform. Theory*, Vol. IT-33, pp. 483-501, July 1987.

[12] G. D. Forney, Jr., "Coset codes I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, Vol. 34, pp. 1123-1151, Sept. 1988, Part II.

[13] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, Vol. 34, pp. 1152-1187, Sept. 1988, Part II.

[14] R. H. Deng and D. J. Costello, Jr., "High rate concatenated coding systems using multi-dimensional bandwidth efficient trellis inner codes," *IEEE Trans. Commun.*, to appear, 1989.

[15] J. P. Odenwalder, "Optimal decoding of convolutional codes," Ph.D Thesis, University of California, Los Angeles, 1970.

[16] H. Imai and S. Hirakawa, "A new multilevel coding method using error correcting codes," *IEEE Trans. Inform. Theory*, Vol. IT-23, pp. 371-377, May 1977.

[17] E. L. Cusack, "Error control codes for QAM signalling," Electronics Letters, Vol. 20, No. 2, pp. 62-63, 19 Jan. 1984.

[18] V. V. Ginzburg, "Multidimensional signals for a continuous channel," *Problemy Peredachi Informatsii* [Problems of Information Transmission], Vol. 20, No.1, pp. 28-46, Jan.-Mar. 1984 (In Russian).

[19] S. I. Sayegh, "A class of optimum block codes in signal space," *IEEE Trans. Commun.*, Vol. COM-34, pp. 1043-1045, Oct. 1986.

[20] M. Rouanne and D. J. Costello, Jr., "A lower bound on the minimum Euclidean distance of trellis coded modulation schemes," *IEEE Trans. Inform. Theory*, Vol. 34, pp. 1011-1020, Sept. 1988, Part I.

[21] E. Zehavi and J. K. Wolf, "On the perfomance evaluation of trellis codes," *IEEE Trans. Inform. Theory*, Vol. IT-33, pp. 196-202, March 1987.

[22] K. J. Larson, 'Comments on "An efficient algorithm for computing free distance",' *IEEE Trans. Inform. Theory*, Vol. IT-18, pp. 437-439, May 1972.

[23] D. Divsalar and M. K. Simon, "Multiple trellis coded modulation (MTCM)," *IEEE Trans. Commun.*, Vol. COM-36, pp. 410-419, April 1988.

[24] F. Hemmati and R. J. F. Fang, "Low complexity coding methods for high data rate channels," *Comsat Technical Review*, Vol. 16, pp. 425-447, Fall 1986.

[25] S. S. Pietrobon, "Rotationally invariant convolutional codes for MPSK modulation and implementation of Viterbi decoders," Masters Thesis, South Australian Institute of Technology, Adelaide, June 1988.

[26] E. Biglieri and M. Elia, "Multidimensional modulation and coding," *IEEE Trans. Inform. Theory*, Vol. 34, pp. 803-809, July 1988.

[27] A. R. Calderbank, "Multilevel codes and multistage decoding," *IEEE Trans. Commun.*, Vol. 37, pp. 222-229, Mar. 1989.

[28] R. M. Tanner, "Algebraic construction of large Euclidean distance combined coding/modulation systems," *IEEE Trans. Inform. Theory*, to appear.

[29] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On construction of bandwidth efficient block codes," *IEEE Trans. Inform. Theory*, to appear.

[30] G. J. Pottie and D. P. Taylor, "Multi-level codes based on partitioning," *IEEE Trans. Inform. Theory*, to appear.

Table 1: 2×8PSK Signal Set Partition

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0, C_0, \underset{\downarrow}{C_0})$	min(4 , 2 , 0 .586) = 0 . 586	[0 1]
1	$\Omega(C_0, C_0, \underset{\downarrow}{C_1})$	min(4 , 2 , 1 .172) = 1 . 172	[1 1]
2	$\Omega(C_0, \underset{\downarrow}{C_0}, C_2)$	min(4 , 2 , ∞) = 2 . 0	[0 2]
3	$\Omega(C_0, \underset{\downarrow}{C_1}, C_2)$	min(4 , 4 , ∞) = 4 . 0	[2 2]
4	$\Omega(\underset{\downarrow}{C_0}, C_2, C_2)$	min(4 , ∞, ∞) = 4 . 0	[0 4]
5	$\Omega(\underset{\downarrow}{C_1}, C_2, C_2)$	min(8 , ∞, ∞) = 8 . 0	[4 4]
6	$\Omega(C_2, C_2, C_2)$	min(∞, ∞, ∞) = ∞	-

Table 2: Binary generators for L = 3 and 4.

(a) L = 3 (I)

m	d_m	N_m	$(\tau_1^m)^T$
0	1	3	[1 1 1]
1	2	3	[1 1 0]
2	2	1	[0 1 1]

(b) L = 3 (II)

m	d_m	N_m	$(\tau_2^m)^T$
0	1	3	[0 0 1]
1	1	1	[0 1 1]
2	3	1	[1 1 1]

(c) L = 4

m	d_m	N_m	$(\tau^m)^T$
0	1	4	[0 0 0 1]
1	2	6	[0 0 1 1]
2	2	2	[0 1 0 1]
3	4	1	[1 1 1 1]

Table 3(a): 3×8PSK Signal Set Partition (I)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0,C_0,\underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[1 1 1]
1	$\Omega(C_0,C_0,\underset{\downarrow}{C_1^1})$	min(4, 2, 1.172) = 1.172	[1 1 0]
2	$\Omega(C_0,C_0,\underset{\downarrow}{C_2^1})$	min(4, 2, 1.172) = 1.172	[0 1 1]
3	$\Omega(C_0,\underset{\downarrow}{C_0},C_3)$	min(4, 2, ∞) = 2.0	[2 2 2]
4	$\Omega(C_0,\underset{\downarrow}{C_1^1},C_3)$	min(4, 4, ∞) = 4.0	[2 2 0]
5	$\Omega(C_0,\underset{\downarrow}{C_2^1},C_3)$	min(4, 4, ∞) = 4.0	[0 2 2]
6	$\Omega(\underset{\downarrow}{C_0},C_3,C_3)$	min(4, ∞, ∞) = 4.0	[4 4 4]
7	$\Omega(\underset{\downarrow}{C_1^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[0 4 4]
9	$\Omega(C_3,C_3,C_3)$	min(∞, ∞, ∞) = ∞	-

Table 3(b): 3×8PSK Signal Set Partition (II)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0,C_0,\underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[0 0 1]
1	$\Omega(C_0,C_0,\underset{\downarrow}{C_1^2})$	min(4, 2, 0.586) = 0.586	[0 1 1]
2	$\Omega(C_0,C_0,\underset{\downarrow}{C_2^2})$	min(4, 2, 1.757) = 1.757	[1 1 1]
3	$\Omega(C_0,\underset{\downarrow}{C_0},C_3)$	min(4, 2, ∞) = 2.0	[2 2 2]
4	$\Omega(C_0,\underset{\downarrow}{C_1^1},C_3)$	min(4, 4, ∞) = 4.0	[2 2 0]
5	$\Omega(C_0,\underset{\downarrow}{C_2^1},C_3)$	min(4, 4, ∞) = 4.0	[0 2 2]
6	$\Omega(\underset{\downarrow}{C_0},C_3,C_3)$	min(4, ∞, ∞) = 4.0	[4 4 4]
7	$\Omega(\underset{\downarrow}{C_1^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1},C_3,C_3)$	min(8, ∞, ∞) = 8.0	[0 4 4]
9	$\Omega(C_3,C_3,C_3)$	min(∞, ∞, ∞) = ∞	-

Table 3(c): 3×8PSK Signal Set Partition (III)

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0, C_0, \underset{\downarrow}{C_0})$	min(4, 2, 0.586) = 0.586	[0 0 1]
1	$\Omega(C_0, C_0, \underset{\downarrow}{C_1^2})$	min(4, 2, 0.586) = 0.586	[0 1 1]
2	$\Omega(C_0, C_0, \underset{\downarrow}{C_2^2})$	min(4, 2, 1.757) = 1.757	[1 1 1]
3	$\Omega(C_0, \underset{\downarrow}{C_0}, C_3)$	min(4, 2, ∞) = 2.0	[0 0 2]
4	$\Omega(C_0, \underset{\downarrow}{C_1^2}, C_3)$	min(4, 2, ∞) = 2.0	[0 2 2]
5	$\Omega(\underset{\downarrow}{C_0}, C_2^2, C_3)$	min(4, 6, ∞) = 4.0	[4 4 4]
6	$\Omega(C_1^1, \underset{\downarrow}{C_2^2}, C_3)$	min(8, 6, ∞) = 6.0	[2 2 2]
7	$\Omega(\underset{\downarrow}{C_1^1}, C_3, C_3)$	min(8, ∞, ∞) = 8.0	[4 4 0]
8	$\Omega(\underset{\downarrow}{C_2^1}, C_3, C_3)$	min(8, ∞, ∞) = 8.0	[0 4 4]
9	$\Omega(C_3, C_3, C_3)$	min(∞, ∞, ∞) = ∞	-

Table 4: 4×8PSK Signal Set Partition.

Partition Level (p)	Ω^p	Minimum squared subset distance (Δ_p^2)	Generator $(t^p)^T$
0	$\Omega(C_0, C_0, C_0)$	min(4, 2, 0.586) = 0.586	[0 0 0 1]
1	$\Omega(C_0, C_0, \overset{\downarrow}{C}_1)$	min(4, 2, 1.172) = 1.172	[0 0 1 1]
2	$\Omega(C_0, C_0, \overset{\downarrow}{C}_2)$	min(4, 2, 1.172) = 1.172	[0 1 0 1]
3	$\Omega(C_0, C_0, \overset{\downarrow}{C}_3)$	min(4, 2, 2.343) = 2.0	[0 0 0 2]
4	$\Omega(C_0, \overset{\downarrow}{C}_1, C_3)$	min(4, 4, 2.343) = 2.343	[1 1 1 1]
5	$\Omega(C_0, C_1, \overset{\downarrow}{C}_4)$	min(4, 4, ∞) = 4.0	[0 0 2 2]
6	$\Omega(C_0, \overset{\downarrow}{C}_2, C_4)$	min(4, 4, ∞) = 4.0	[0 2 0 2]
7	$\Omega(C_0, \overset{\downarrow}{C}_3, C_4)$	min(4, 8, ∞) = 4.0	[0 0 0 4]
8	$\Omega(\overset{\downarrow}{C}_1, C_3, C_4)$	min(8, 8, ∞) = 8.0	[2 2 2 2]
9	$\Omega(C_1, \overset{\downarrow}{C}_4, C_4)$	min(8, ∞, ∞) = 8.0	[0 0 4 4]
10	$\Omega(\overset{\downarrow}{C}_2, C_4, C_4)$	min(8, ∞, ∞) = 8.0	[0 4 0 4]
11	$\Omega(\overset{\downarrow}{C}_3, C_4, C_4)$	min(16, ∞, ∞) =16.0	[4 4 4 4]
12	$\Omega(\overset{\downarrow}{C}_4, C_4, C_4)$	min(∞, ∞, ∞) = ∞	-

Table 5: Summary of L×4PSK partitions.

	L = 2		L=3 (I)		L=3'(II)		L=3 (III)		L = 4	
Partition Level (p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)
0	2	01	2	111	2	001	2	001	2	0001
1	4	11	4	110	2	011	2	011	4	0011
2	4	02	4	011	4	222	4	002	4	0101
3	8	22	4	222	6	111	4	022	4	0002
4	-	-	8	220	8	220	6	111	8	1111
5	-	-	8	022	8	022	12	222	8	0022
6	-	-	-	-	-	-	-	-	8	0202
7	-	-	-	-	-	-	-	-	16	2222
p_0 p_1	1 3		0 3		3 2		4 5		4 7	

Table 6: Summary of L×8PSK partitions.

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)	MSSD (Δ_p^2)	gen. (t^p)
0	0.586	01	0.586	111	0.586	001	0.586	001	0.586	0001
1	1.172	11	1.172	110	0.586	011	0.586	011	1.172	0011
2	2	02	1.172	011	1.757	111	1.757	111	1.172	0101
3	4	22	2	222	2	222	2	002	2	0002
4	4	04	4	220	4	220	2	022	2.343	1111
5	8	44	4	022	4	022	4	444	4	0022
6	-	-	4	444	4	444	6	222	4	0202
7	-	-	8	440	8	440	8	440	4	0004
8	-	-	8	044	8	044	8	044	8	2222
9	-	-	-	-	-	-	-	-	8	0044
10	-	-	-	-	-	-	-	-	8	0404
11	-	-	-	-	-	-	-	-	16	4444
p_0 p_1 p_2	1 3 5		0 3 6		2 3 6		2 6 5		4 8 11	

Table 7: Summary of L×16PSK partitions.

	L = 2		L=3 (I)		L=3 (II)		L=3 (III)		L = 4	
Partition Level (p)	MSSD (Δ_p^2)	gen. (t^{pT})	MSSD (Δ_p^2)	gen. (t^{pT})	MSSD (Δ_p^2)	gen. (t^{pT})	MSSD (Δ_p^2)	gen. (t^{pT})	MSSD (Δ_p^2)	gen. (t^{pT})
0	0.152	01	0.152	111	0.152	001	0.152	001	0.152	0001
1	0.304	11	0.304	110	0.152	011	0.152	011	0.304	0011
2	0.586	02	0.304	011	0.457	111	0.457	111	0.304	0101
3	1.172	22	0.586	222	0.586	222	0.586	002	0.586	0002
4	2	04	1.172	220	1.172	220	0.586	022	0.609	1111
5	4	44	1.172	022	1.172	022	1.757	222	1.172	0022
6	4	08	2	444	2	444	2	444	1.172	0202
7	8	88	4	440	4	440	4	440	2	0004
8	-	-	4	044	4	044	4	044	2.343	2222
9	-	-	4	888	4	888	4	888	4	0044
10	-	-	8	880	8	880	8	880	4	0404
11	-	-	8	088	8	088	8	088	4	0008
12	-	-	-	-	-	-	-	-	8	4444
13	-	-	-	-	-	-	-	-	8	0088
14	-	-	-	-	-	-	-	-	8	0808
15	-	-	-	-	-	-	-	-	16	8888
$p_0p_1p_2p_3$	1 3 5 7		0 3 6 9		2 3 6 9		2 5 6 9		4 8 12 15	

Table 8: 2×2×8PSK signal set partition.

p	Ω^p	Minimum squared subset distance (Δ_p^2)		gen. $(t^p)^T$
0	$\Omega(C_0,C_0,C_0,C_0,C_0,C_0)$	min(8, 4, 4, 2, 1.172, 0.586)	= 0.586	[0 1]
1	$\Omega(C_0,C_0,C_0,C_0,C_0,\overset{\downarrow}{C}_1)$	min(8, 4, 4, 2, 1.172, 1.172)	= 1.172	[1 1]
2	$\Omega(C_0,C_0,C_0,C_0,C_0,\overset{\downarrow}{C}_2)$	min(8, 4, 4, 2, 1.172, ∞)	= 1.172	[0 2]
3	$\Omega(C_0,C_0,C_0,C_0,\overset{\downarrow}{C}_1,C_2)$	min(8, 4, 4, 2, 2.343, ∞)	= 2.0	[0 4]
4	$\Omega(C_0,C_0,C_0,\overset{\downarrow}{C}_1,C_1,C_2)$	min(8, 4, 4, 4, 2.343, ∞)	= 2.343	[2 2]
5	$\Omega(C_0,C_0,C_0,C_1,\overset{\downarrow}{C}_2,C_2)$	min(8, 4, 4, 4, ∞, ∞)	= 4.0	[4 4]
6	$\Omega(C_0,C_0,C_0,\overset{\downarrow}{C}_2,C_2,C_2)$	min(8, 4, 4, ∞, ∞, ∞)	= 4.0	[0 8]
7	$\Omega(C_0,C_0,\overset{\downarrow}{C}_1,C_2,C_2,C_2)$	min(8, 4, 8, ∞, ∞, ∞)	= 4.0	[0 16]
8	$\Omega(C_0,\overset{\downarrow}{C}_1,C_1,C_2,C_2,C_2)$	min(8, 8, 8. ∞, ∞, ∞)	= 8.0	[8 8]
9	$\Omega(C_0,C_1,\overset{\downarrow}{C}_2,C_2,C_2,C_2)$	min(8, 8, ∞, ∞, ∞, ∞)	= 8.0	[16 16]
10	$\Omega(C_0,\overset{\downarrow}{C}_2,C_2,C_2,C_2,C_2)$	min(8, ∞, ∞, ∞, ∞, ∞)	= 8.0	[0 32]
11	$\Omega(\overset{\downarrow}{C}_1,C_2,C_2,C_2,C_2,C_2)$	min(16, ∞, ∞, ∞, ∞, ∞)	=16.0	[32 32]
12	$\Omega(\overset{\downarrow}{C}_2,C_2,C_2,C_2,C_2,C_2)$	min(∞, ∞, ∞, ∞, ∞, ∞)	= ∞	-

Table 9: Binary generators for L = 5 and 7.

	L = 5 (I)			L = 5 (II)			L = 5 (III)		
m	d_m	N_m	$(\tau_1^m)^T$	d_m	N_m	$(\tau_2^m)^T$	d_m	N_m	$(\tau_3^m)^T$
0	1	5	[11111]	1	5	[11111]	1	5	[00001]
1	2	10	[00011]	1	2	[00001]	1	1	[00010]
2	2	4	[00101]	2	2	[00110]	2	3	[00101]
3	2	1	[11000]	3	2	[10101]	2	1	[01001]
4	4	1	[01111]	4	1	[01111]	5	1	[11111]

	L = 7 (I)			L = 7 (II)			L = 7 (III)			L = 7 (IV)		
m	d_m	N_m	$(\tau_1^m)^T$	d_m	N_m	$(\tau_2^m)^T$	d_m	N_m	$(\tau_3^m)^T$	d_m	N_m	$(\tau_4^m)^T$
0	1	7	[1111111]	1	7	[1111111]	1	7	[0000001]	1	7	[0000001]
1	2	21	[0000011]	1	2	[0000001]	1	1	[0001000]	1	3	[0001000]
2	2	9	[0001001]	2	5	[0000101]	2	6	[1111111]	1	1	[1000000]
3	2	3	[0010010]	2	1	[0100010]	2	2	[0000101]	3	7	[0110100]
4	2	1	[0001100]	3	3	[0011100]	3	2	[0101010]	3	3	[0011010]
5	4	2	[1111000]	4	2	[0001111]	4	1	[1100011]	3	1	[0001101]
6	6	1	[0111111]	6	1	[1110111]	5	1	[0011111]	7	1	[1111111]

Table 10: Squared Euclidean Weights used in the code search for rate 7/8 (2.33 bit/T) codes with 3×8PSK (II) and $\tilde{k} = 2$.

$\tilde{e}_n$	$w^2(\tilde{e}_n)$	$m(\tilde{e}_n)$
000	0.0	1
001	1.172	2
010	1.757	4
011	0.586	1
100	2.0	6
101	1.172	2
110	1.757	4
111	0.586	1

Table 11: Trellis Coded 1×4PSK.

$R_{eff} = 1.0$ bit/T, $d_u^2 = 4.0$, $N_u = 1$ (1×2PSK).

ν	$\tilde{k}$	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	1	3	360°	6	1	-	-	1.76
2	1	2	5	360°	10	1	-	-	3.98
3	1	06	13	180°	12	2	-	-	4.77
	1	04	13	360°	12	1	-	-	4.77
4	1	06	21	180°	12	1	-	-	4.77
	1	10	23	360°	14	2	-	-	5.44
5	1	36	45	180°	16	2	-	-	6.02
	1	26	53	360°	16	1	-	-	6.02
6	1	042	117	180°	20	11	-	-	6.99
7	1	126	235	180°	20	2	-	-	6.99
	1	144	223	360°	20	1	-	-	6.99
8	1	262	435	180°	24	11	-	-	7.78
	1	362	515	360°	24	9	-	-	7.78
9	1	0644	1123	180°	24	2	-	-	7.78
	1	0712	1047	360°	24	1	-	-	7.78

$\gamma_2 = 0$ dB

Table 12(a): Trellis Coded 2×4PSK.

R_{eff} = 1.5 bit/T, q=0, d_u^2 = 4, N_u = 6 (2×4PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	180^o	4	2	6	8	0.00
2	2	-	1	3	5	90^o	6	6	-	-	1.76
3	2	-	04	06	11	90^o	8	5	-	-	3.01
4	2	-	10	06	23	90^o	8	1	10	16	3.01
5	3	14	30	02	41	180^o	10	8	-	-	3.98
	3	16	24	06	53	360^o	10	7	-	-	3.98
6	3	030	042	014	103	180^o	12	40.25	-	-	4.77
	3	076	024	010	157	360^o	12	30.75	-	-	4.77
7	3	044	022	114	211	180^o	12	8	-	-	4.77

γ_2 = 1.76 dB

Table 12(b): Trellis Coded 2×4PSK.

R_{eff} = 1.0 bit/T, q=1, d_u^2 = 4.0, N_u = 1 (1×2PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	90^o	8	5	-	-	3.01
2	1	-	2	5	90^o	8	1	12	8	3.01
3	2	04	02	11	360^o	12	5	-	-	4.77
4	2	14	06	23	180^o	12	1	-	-	4.77
5	2	30	16	41	180^o	16	8	-	-	6.02
6	2	036	052	115	180^o	16	1	-	-	6.02
7	2	044	136	203	180^o	20	6	-	-	6.99
8	2	110	226	433	180^o	24	33	-	-	7.78

γ_2 = 0 dB

Table 13(a): Trellis Coded 3×4PSK.

R_{eff} = 1.67 bit/T, q=0, d_u^2 = 4.0, N_u = 15 (3×4PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	4	7	6	32	0.00	I
2	2	-	2	1	5	90°	4	3	6	24	0.00	I
	2	-	2	1	5	360°	4	2	-	-	0.00	II
3	2	-	04	02	11	90°	4	1	6	6	0.00	III
	2	-	04	02	11	360°	6	11	-	-	1.76	II
	3	05	04	02	11	90°	4	0.25	-	-	0.00	III
4	2	-	14	02	21	180°	6	6	-	-	1.76	II
3	3	01	02	06	11	360°	6	4	-	-	1.76	II
4	3	10	04	02	21	90°	6	5.5	-	-	1.76	III
	3	12	04	02	21	180°	8	19	-	-	3.01	I
5	3	24	14	02	41	180°	8	7	-	-	3.01	I
6	3	024	042	010	105	180°	8	3	10	16	3.01	I

γ_2 = 2.22 dB

Table 13(b): Trellis Coded 3×4PSK.

R_{eff} = 1.33 bit/T, q=1, d_u^2 = 4.0, N_u = 3 (3×4PSK II).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	4	1	8	4	0.00	III
	1	-	-	1	3	360°	6	7	-	-	1.76	II
2	1	-	-	2	5	360°	6	4	10	9	1.76	II
	2	-	2	1	5	90°	6	2	8	4	1.76	III
	2	-	3	1	5	180°	8	21	-	-	3.01	I
	2	-	2	1	5	360°	8	16	-	-	3.01	II
3	2	-	04	02	11	90°	6	2	8	1	1.76	III
	2	-	02	06	11	180°	8	3	12	100	3.01	II
	3	06	04	03	11	90°	8	1	-	-	3.01	III
4	3	14	04	12	23	90°	10	5	-	-	3.98	III
5	3	30	04	22	43	90°	12	13	-	-	4.77	III
6	3	036	060	026	103	90°	12	2	-	-	4.77	III
7	3	140	160	062	213	90°	12	1	14	5	4.77	III
	3	004	154	056	207	180°	12	1	16	128	4.77	III

γ_2 = 1.25 dB

Table 13(c): Trellis Coded 3×4PSK.

R_{eff} = 1.00 bit/T, q=2, d_u^2 = 4.0, N_u = 1 (1×2PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d_{free}^2	N_{free}	d_{next}^2	N_{next}	γ (dB)	sig. set
0	0	-	-	-	-	90°	6	4	-	-	1.76	II
1	1	-	-	1	3	90°	6	2	8	1	1.76	III
	1	-	-	1	3	180°	8	3	12	16	3.01	II
2	2	-	3	2	5	90°	10	4	-	-	3.98	III
3	2	-	06	02	11	90°	10	2	-	-	3.98	III
	2	-	02	06	13	180°	12	5	-	-	4.77	III
4	2	-	12	16	21	90°	12	1	14	2	4.77	III
	2	-	04	12	27	180°	12	1	16	22	4.77	III
	3	10	04	02	21	180°	14	3	-	-	5.44	II
5	3	22	16	04	53	180°	16	2	-	-	6.02	II
	3	24	14	02	43	360°	16	1	-	-	6.02	II
6	3	070	004	022	101	180°	18	3	-	-	6.53	II
7	3	156	024	046	213	180°	20	3	-	-	6.99	II
	3	044	014	102	217	360°	20	2	-	-	6.99	II

γ_2 = 0.0 dB

Table 14(a): Trellis Coded 4×4PSK.

R_{eff} = 1.75 bit/T, q=0, d_u^2 = 4.0, N_u = 28 (4×4PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	4	12	6	64	0.00
2	2	-	2	1	5	90°	4	4	6	48	0.00
3	3	04	02	01	11	90°	6	28	-	-	1.76
4	3	10	04	02	21	90°	8	78	-	-	3.01
5	3	24	14	02	41	90°	8	30	-	-	3.01
6	3	050	032	004	103	90°	8	14	10	160	3.01

γ_2 = 2.43 dB

Table 14(b): Trellis Coded 4×4PSK.

R_{eff} = 1.50 bit/T, q=1, d_u^2 = 4.0, N_u = 6 (2×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	4	4	8	64	0.00
2	2	-	-	2	1	5	90°	8	78	-	-	3.01
3	2	-	-	04	02	11	90°	8	30	-	-	3.01
4	2	-	-	12	04	23	90°	8	16	12	320	3.01
5	3	-	14	34	06	41	90°	8	6	12	176	3.01
	3	-	04	14	22	43	180°	8	6	12	160	3.01
6	4	014	006	056	022	103	90°	8	2	12	62	3.01

γ_2 = 1.76 dB

Table 14(c): Trellis Coded 4×4PSK.

R_{eff} = 1.25 bit/T, q=2, d_u^2 = 4.0, N_u = 4 (4×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	8	30	-	-	3.01
2	1	-	-	-	2	5	90°	8	14	12	64	3.01
3	2	-	-	06	02	11	90°	8	6	12	64	3.01
	2	-	-	02	06	11	180°	8	6	12	32	3.01
	3	-	01	03	06	11	90°	8	2	12	56	3.01
4	3	-	10	14	06	21	90°	8	2	12	8	3.01
5	4	10	04	06	22	41	90°	12	8	-	-	4.77
6	4	024	014	006	042	103	90°	16	109	-	-	6.02

γ_2 = 0.97 dB

Table 14(d): Trellis Coded 4×4PSK.

R_{eff} = 1.00 bit/T, q=3, d_u^2 = 4.0, N_u = 1 (1×2PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
0	0	-	-	-	-	90°	8	14	-	-	3.01
1	1	-	-	1	3	180°	8	6	16	64	3.01
2	2	-	2	3	5	90°	8	2	16	64	3.01
3	3	02	04	03	11	90°	16	45	-	-	6.02
4	3	02	10	06	21	90°	16	17	-	-	6.02
5	3	22	10	06	41	90°	16	5	-	-	6.02
6	3	010	060	036	105	90°	16	1	20	4	6.02

γ_2 = 0 dB

Table 15: Trellis Coded 1×8PSK.

R_{eff} = 2.0 bit/T, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	180°	2.586	2	-	-	1.12
2	1	-	2	5	180°	4.0	1	4.586	4	3.01
3	2	04	02	11	360°	4.586	2	-	-	3.60
4	2	14	06	23	180°	5.172	4	-	-	4.13
	2	16	04	23	360°	5.172	2.25	-	-	4.13
5	2	14	26	53	180°	5.172	0.25	-	-	4.13
	2	20	10	45	360°	5.757	2	-	-	4.59
6	2	074	012	147	180°	6.343	3.25	-	-	5.01
7	2	146	052	225	180°	6.343	0.125	-	-	5.01
	2	122	054	277	360°	6.586	0.5	-	-	5.18
8	2	146	210	573	180°	7.515	3.375	-	-	5.75
	2	130	072	435	360°	7.515	1.5	-	-	5.75

γ_4 = 0 dB

Table 16(a): Trellis Coded 2×8PSK.

R_{eff} = 2.5 bit/T, q=0, d_u^2 = 1.172, N_u = 4 (2×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	90°	1.757	8	2.0	4	1.76
2	1	-	-	2	5	90°	2.0	4	2.929	32	2.32
3	2	-	04	06	11	45°	2.929	16	-	-	3.98
4	2	-	16	12	23	45°	3.515	56	-	-	4.77
5	2	-	10	06	41	45°	3.515	16	-	-	4.77
6	2	-	004	030	113	45°	4.0	6	4.101	80	5.33
	2	-	044	016	107	90°	4.0	6	4.101	48	5.33
7	3	110	044	016	317	90°	4.0	2	4.101	25	5.33

γ_4 = -1.35 dB

Table 16(b): Trellis Coded 2×8PSK.

R_{eff} = 2.0 bit/T, q=1, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45°	3.172	8	4.0	6	2.00
2	1	-	-	2	5	45°	4.0	6	5.172	32	3.01
3	2	-	04	02	11	180°	4.0	2	5.172	16	3.01
4	3	04	14	02	21	90°	5.172	8	-	-	4.13
5	3	24	14	06	43	90°	6.0	6	-	-	4.77
6	3	012	050	004	125	90°	6.343	5.5	-	-	5.01
7	3	110	044	016	317	90°	7.515	25	-	-	5.75

γ_4 = 0 dB

Table 17(a): Trellis Coded 3×8PSK.

R_{eff} = 2.67 bit/T, q=0, d_u^2 = 1.172, N_u = 12 (3×8PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	45°	1.172	4	-	-	0.00	II
2	1	-	-	2	5	45°	1.757	16	-	-	1.76	II
3	2	-	04	02	11	45°	2.0	6	2.343	16	2.32	I
4	3	14	04	02	21	90°	2.343	12	-	-	3.01	I
	3	10	04	02	21	180°	2.343	8	-	-	3.01	I
5	3	30	14	02	53	90°	2.929	48	-	-	3.98	I
6	3	050	022	006	103	90°	3.172	12	-	-	4.33	I
7	3	056	112	004	225	90°	3.515	84	-	-	4.77	I
	3	100	050	022	255	180°	3.515	76	-	-	4.77	I

γ_4 = -1.07 dB

Table 17(b): Trellis Coded 3×8PSK.

R_{eff} = 2.33 bit/T, q=1, d_u^2 = 1.757, N_u = 8 (3×8PSK II).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	90°	2.0	6	2.343	16	0.56	II
2	2	-	-	3	1	7	90°	2.586	6	-	-	1.68	II
3	2	-	-	06	02	11	90°	3.515	16	-	-	3.01	II
	2	-	-	04	02	11	180°	3.757	24	-	-	3.30	II
4	3	-	10	04	06	21	45°	3.757	12	-	-	3.30	III
	2	-	-	14	02	27	90°	4.0	15	4.343	24	3.57	II
5	3	-	22	16	06	41	45°	4.0	7	-	-	3.57	III
6	3	-	010	046	060	105	45°	4.0	3	4.686	8	3.57	III
	4	060	024	014	002	101	180°	4.0	2	-	-	3.57	III

γ_4 = 0.11 dB

Table 17(c): Trellis Coded 3×8PSK.

R_{eff} = 2.00 bit/T, q=2, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	-	1	3	180°	3.757	24	-	-	2.74	II
2	1	-	-	-	2	5	180°	4.0	15	5.757	144	3.01	II
3	2	-	-	04	02	11	45°	4.0	7	-	-	3.01	III
4	2	-	-	12	04	27	45°	4.0	3	5.757	32	3.01	III
5	3	-	14	24	02	41	180°	5.757	17.5	-	-	4.59	III
	3	-	16	22	06	53	360°	5.757	17	-	-	4.59	III
6	3	-	030	042	014	103	180°	6.0	11	-	-	4.77	III
	4	014	044	024	006	103	180°	6.0	4	-	-	4.77	II

γ_4 = 0 dB

Table 18(a): Trellis Coded 4×8PSK.

R_{eff} = 2.75 bit/T, q=0, d_u^2 = 1.172, N_u = 24 (4×8PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	45^o	1.172	8	1.757	64	0.00
2	2	-	-	2	1	5	45^o	1.757	48	-	-	1.76
3	2	-	-	04	02	11	45^o	2.0	8	2.343	64	2.32
4	3	-	10	04	02	21	45^o	2.343	40	-	-	3.01
5	3	-	30	14	02	41	45^o	2.343	8	2.929	288	3.01
6	4	030	020	052	014	101	45^o	2.929	136	-	-	3.98

γ_4 = -0.94 dB

Table 18(b): Trellis Coded 4×8PSK.

R_{eff} = 2.50 bit/T, q=1, d_u^2 = 1.172, N_u = 4 (2×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45^o	2.0	8	2.343	64	2.32
2	2	-	2	1	5	45^o	2.343	40	-	-	3.01
3	2	-	04	02	11	45^o	2.343	8	3.172	32	3.01
4	3	14	04	02	21	45^o	3.172	16	-	-	4.33
5	3	24	14	02	41	45^o	3.515	64	-	-	4.77
6	3	014	024	042	103	45^o	4.0	28	4.686	1088	5.33

γ_4 = -1.35 dB

Table 18(c): Trellis Coded 4×8PSK.

R_{eff} = 2.25 bit/T, q=2, d_u^2 = 2.0, N_u = 8 (4×8PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	45°	2.343	8	3.172	32	0.69
2	2	-	-	3	1	5	45°	3.172	16	-	-	2.00
3	2	-	-	06	02	11	45°	4.0	28	4.343	64	3.01
	2	-	-	02	06	11	90°	4.0	28	4.686	64	3.01
4	3	-	04	06	12	21	45°	4.0	12	4.686	32	3.01
5	4	10	04	06	22	41	45°	4.0	4	4.686	16	3.01

γ_4 = 0.51 dB

Table 18(d): Trellis Coded 4×8PSK.

R_{eff} = 2.00 bit/T, q=3, d_u^2 = 2.0, N_u = 2 (1×4PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	90°	4.0	28	4.686	64	3.01
2	2	-	-	2	3	5	45°	4.0	12	4.686	32	3.01
3	3	-	02	04	03	11	45°	4.0	4	4.686	16	3.01
4	4	10	04	02	03	21	45°	4.686	8	-	-	3.70
5	4	02	10	04	22	41	45°	6.343	16	-	-	5.01
6	4	034	044	016	036	107	45°	6.686	6	-	-	5.24
	4	044	024	014	016	103	90°	7.029	24	-	-	5.46

γ_4 = 0 dB

Table 19: Trellis Coded 1×16PSK

$R_{eff} = 3.0$ bit/T, $d_u^2 = 0.586$, $N_u = 2$ (1×8PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	90°	0.738	2	-	-	1.00
2	1	-	2	5	90°	1.324	4	-	-	3.54
3	1	-	06	13	45°	1.476	8	-	-	4.01
	1	-	04	13	90°	1.476	4	-	-	4.01
4	1	-	06	21	45°	1.476	4	-	-	4.01
	1	-	10	23	90°	1.628	4	-	-	4.44
5	1	-	24	43	45°	1.781	8	-	-	4.83
	1	-	10	45	90°	1.910	8	-	-	5.13
6	1	-	056	135	45°	2.0	2	2.085	16	5.33
	1	-	032	107	90°	2.0	2	2.085	8	5.33
7	1	-	126	235	45°	2.0	2	2.366	16	5.33
8	2	344	162	717	90°	2.085	2.938	-	-	5.51
	2	224	112	527	180°	2.085	1.219	-	-	5.51

$\gamma_8 = 0$ dB

Table 20(a): Trellis Coded 2×16PSK.

$R_{eff} = 3.5$ bit/T, q=0, $d_u^2 = 0.304$, $N_u = 4$ (2×16PSK).

ν	$\tilde{k}$	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	1	3	45°	0.457	8	-	-	1.76
2	1	-	2	5	45°	0.586	4	0.761	32	2.84
3	2	04	06	11	22.5°	0.761	16	-	-	3.98
4	2	16	12	23	22.5°	0.913	56	-	-	4.77
5	2	10	06	41	22.5°	0.913	16	-	-	4.77
6	2	004	030	113	22.5°	1.066	80	-	-	5.44
	2	044	016	107	45°	1.066	48	-	-	5.44
7	2	074	132	217	22.5°	1.172	4	1.218	228	5.85

$\gamma_8 = -2.17$ dB

Table 20(b): Trellis Coded 2×16PSK.

$R_{eff} = 3.0$ bit/T, q=1, $d_u^2 = 0.586$, $N_u = 2$ (1×8PSK)

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	22.5°	0.890	8	-	-	1.82
2	1	-	-	2	5	22.5°	1.172	4	1.476	32	3.01
3	2	-	04	02	11	90°	1.476	16	-	-	4.01
4	2	-	14	06	23	45°	1.757	8	-	-	4.77
5	2	-	30	16	41	45°	1.781	16	-	-	4.83
6	2	-	044	016	107	45°	2.0	4	2.085	48	5.33
7	3	110	044	016	317	45°	2.085	25	-	-	5.51

$\gamma_8 = 0$ dB

Table 21(a): Trellis Coded 3×16PSK.

$R_{eff} = 3.67$ bit/T, q=0, $d_u^2 = 0.304$, $N_u = 12$ (3×16PSK I).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	22.5°	0.304	4	-	-	0.00	II
2	1	-	-	2	5	22.5°	0.457	16	-	-	1.76	II
3	2	-	04	02	11	22.5°	0.586	6	0.609	16	2.84	I
4	3	14	04	02	21	45°	0.609	12	-	-	3.01	I
	3	10	04	02	21	90°	0.609	8	-	-	3.01	I
5	3	30	14	02	53	45°	0.761	48	-	-	3.98	I
6	3	050	022	006	103	45°	0.890	12	-	-	4.66	I
7	3	056	112	004	225	45°	0.913	84	-	-	4.77	I
	3	100	050	022	255	90°	0.913	76	-	-	4.77	I

$\gamma_8 = 0$ dB

Table 21(b): Trellis Coded 3×16PSK.

$R_{eff} = 3.33$ bit/T, q=1, $d_u^2 = 0.457$, $N_u = 8$ (3×16PSK II).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	45°	0.586	6	0.609	16	1.08	II
2	2	-	3	1	7	45°	0.738	6	-	-	2.08	II
3	2	-	06	02	11	45°	0.913	16	-	-	3.01	II
	2	-	04	02	11	90°	1.043	24	-	-	3.58	II
4	3	10	04	06	21	22.5°	1.043	12	-	-	3.58	III
	2	-	14	02	27	45°	1.172	12	1.195	24	4.09	II
5	3	34	16	06	41	22.5°	1.172	4	-	-	4.09	III
6	3	032	046	006	103	22.5°	1.218	8	-	-	4.26	III
7	3	014	102	044	203	22.5°	1.370	32	-	-	4.77	III
	3	006	072	062	223	45°	1.476	8	-	-	5.09	III

$\gamma_8 = -1.97$ dB

Table 21(c): Trellis Coded 3×16PSK.

R_{eff} = 3.00 bit/T, q=2, d_u^2 = 0.586, N_u = 2 (1×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)	sig. set
1	1	-	-	1	3	90°	1.043	24	-	-	2.50	II
2	1	-	-	2	5	90°	1.172	12	1.628	144	3.01	II
3	2	-	04	02	11	22.5°	1.172	4	-	-	3.01	III
4	2	-	12	04	27	22.5°	1.628	32	-	-	4.44	III
5	2	-	14	02	41	22.5°	1.628	16	-	-	4.44	III
	2	-	22	14	43	45°	1.757	16	-	-	4.77	III
6	2	-	054	020	115	22.5°	1.757	8	2.085	48	4.77	III
	3	020	004	012	101	45°	2.0	6	2.085	72	5.33	II
	3	050	030	026	101	90°	2.0	6	2.085	60	5.33	II
7	3	060	106	050	213	45°	2.0	6	2.214	56	5.33	III
	3	016	110	052	203	90°	2.0	6	2.343	64	5.33	III

γ_8 = 0 dB

Table 22(a): Trellis Coded 4×16PSK.

R_{eff} = 3.75 bit/T, q=0, d_u^2 = 0.304, N_u = 24 (4×16PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	22.5°	0.304	8	0.457	64	0.00
2	2	-	-	2	1	5	22.5°	0.457	48	-	-	1.76
3	2	-	-	04	02	11	22.5°	0.586	8	0.609	64	2.84
4	3	-	10	04	02	21	22.5°	0.609	40	-	-	3.01
5	3	-	30	14	02	41	22.5°	0.609	8	0.761	288	3.01
6	4	030	020	052	014	101	22.5°	0.761	136	-	-	3.98

γ_8 = -1.87 dB

Table 22(b): Trellis Coded 4×16PSK.

R_{eff} = 3.50 bit/T, q=1, d_u^2 = 0.304, N_u = 4 (2×16PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	22.5°	0.586	8	0.609	64	2.84
2	2	-	2	1	5	22.5°	0.609	40	-	-	3.01
3	2	-	04	02	11	22.5°	0.609	8	0.890	32	3.01
4	3	14	04	02	21	22.5°	0.890	16	-	-	4.66
5	3	24	14	02	41	22.5°	0.913	64	-	-	4.77
6	3	014	024	042	103	22.5°	1.172	24	1.218	1088	5.85

γ_8 = -2.17 dB

Table 22(c): Trellis Coded 4×16PSK.

R_{eff} = 3.25 bit/T, q=2, d_u^2 = 0.586, N_u = 8 (4×16PSK).

ν	$\tilde{k}$	h^4	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	-	1	3	22.5°	0.609	8	0.890	32	0.17
2	2	-	-	3	1	5	22.5°	0.890	16	-	-	1.82
3	2	-	-	06	02	11	22.5°	1.172	24	1.195	64	3.01
	2	-	-	02	06	11	22.5°	1.172	24	1.218	64	3.01
4	3	-	04	06	12	21	22.5°	1.172	8	1.218	32	3.01
5	4	10	04	06	22	41	22.5°	1.218	16	-	-	3.18
6	4	050	030	024	016	101	22.5°	1.499	72	-	-	4.08

γ_8 = 0.35 dB

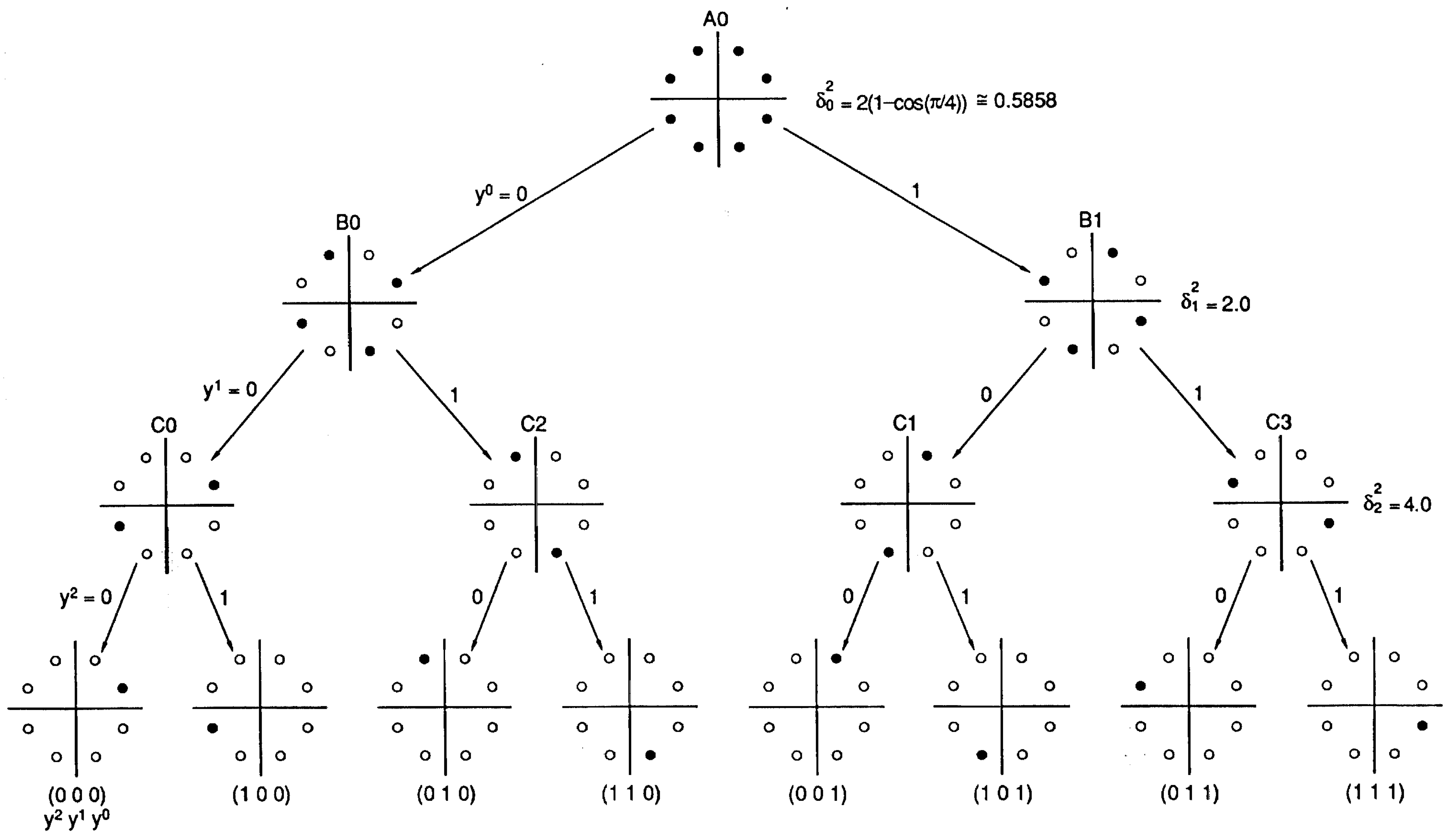
Table 22(d): Trellis Coded 4×16PSK.

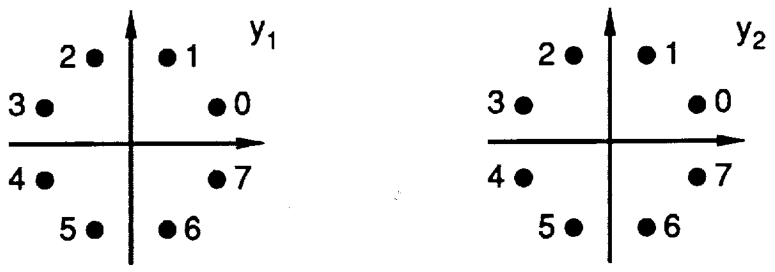
R_{eff} = 3.00 bit/T, q=3, d_u^2 = 0.586, N_u = 2 (1×8PSK).

ν	$\tilde{k}$	h^3	h^2	h^1	h^0	Inv.	d^2_{free}	N_{free}	d^2_{next}	N_{next}	γ (dB)
1	1	-	-	1	3	45°	1.172	24	1.218	64	3.01
2	2	-	2	3	5	22.5°	1.172	8	1.218	32	3.01
3	3	02	04	03	11	22.5°	1.218	16	-	-	3.18
4	3	04	10	06	21	22.5°	1.781	48	-	-	4.83
5	3	22	16	06	41	22.5°	1.804	24	-	-	4.88
	3	24	14	02	43	45°	1.827	64	-	-	4.94
6	3	050	024	006	103	22.5°	2.0	8	2.343	64	5.33

γ_8 = 0 dB

Figure Headings


A0
$\delta_0^2 = 2(1-\cos(\pi/4)) \cong 0.5858$
$y^0 = 0$
1
B0
B1
$\delta_1^2 = 2.0$
$y^1 = 0$
1
0
1
C0
C2
C1
C3
$\delta_2^2 = 4.0$
$y^2 = 0$
1
0
1
0
1
0
1
(0 0 0)
$y^2\ y^1\ y^0$
(1 0 0)
(0 1 0)
(1 1 0)
(0 0 1)
(1 0 1)
(0 1 1)
(1 1 1)

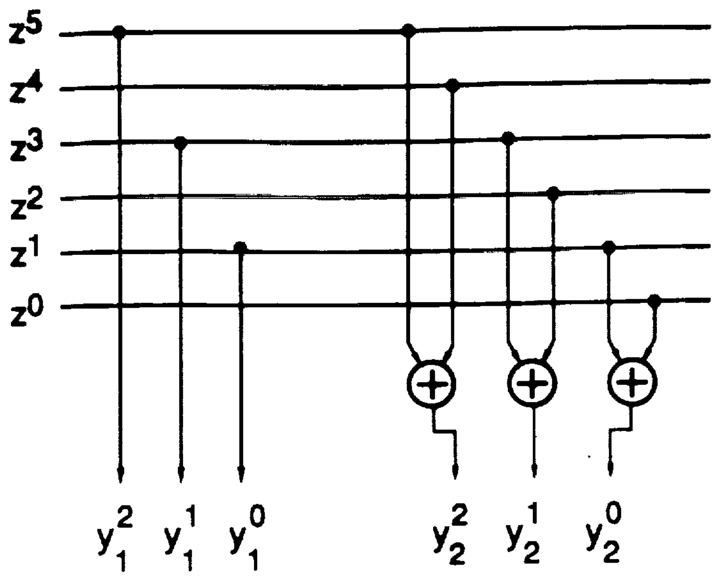


y_1
2
1
3
0
4
7
5
6
y_2
2
1
3
0
4
7
5
6

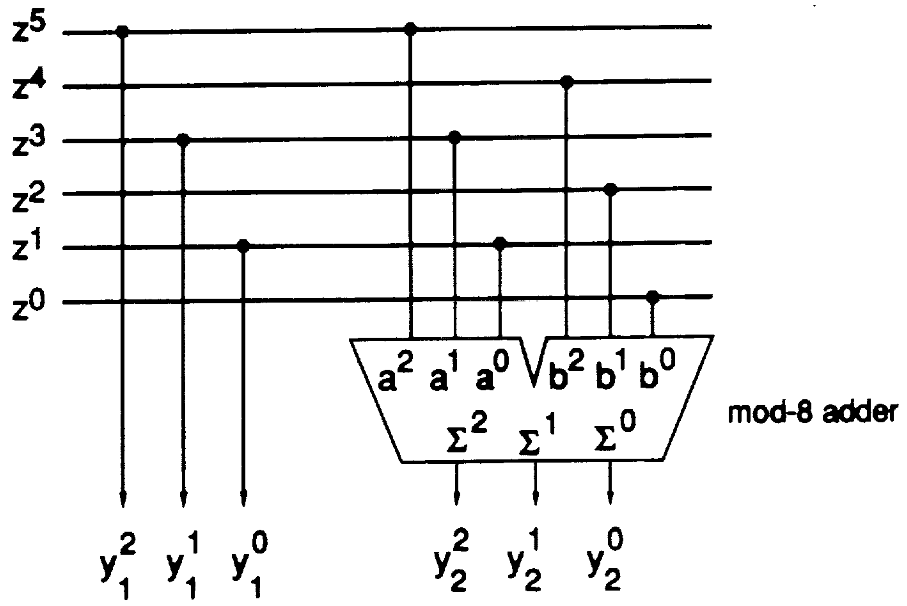
$$\begin{tikzcd}[column sep=large, row sep=small]
 & & C_2(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
 & C_1(0) = C_1 \arrow[ur] \arrow[dr] & \\
 & & C_2(2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
C_0(0) = C_0 \arrow[uur] \arrow[ddr] & & \\
 & & C_2(1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
 & C_1(1) = C_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \arrow[ur] \arrow[dr] & \\
 & & C_2(3) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{tikzcd}$$

$m = 0$ $m = 1$ $m = 2$

$d_0 = 1$ $d_1 = 2$ $d_2 = \infty$


z^5
z^4
z^3
z^2
z^1
z^0
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0

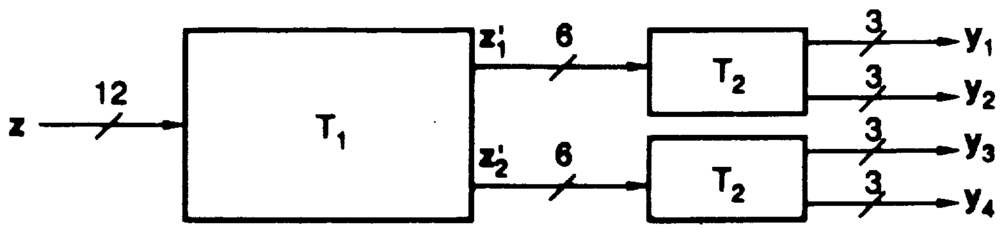
(a)

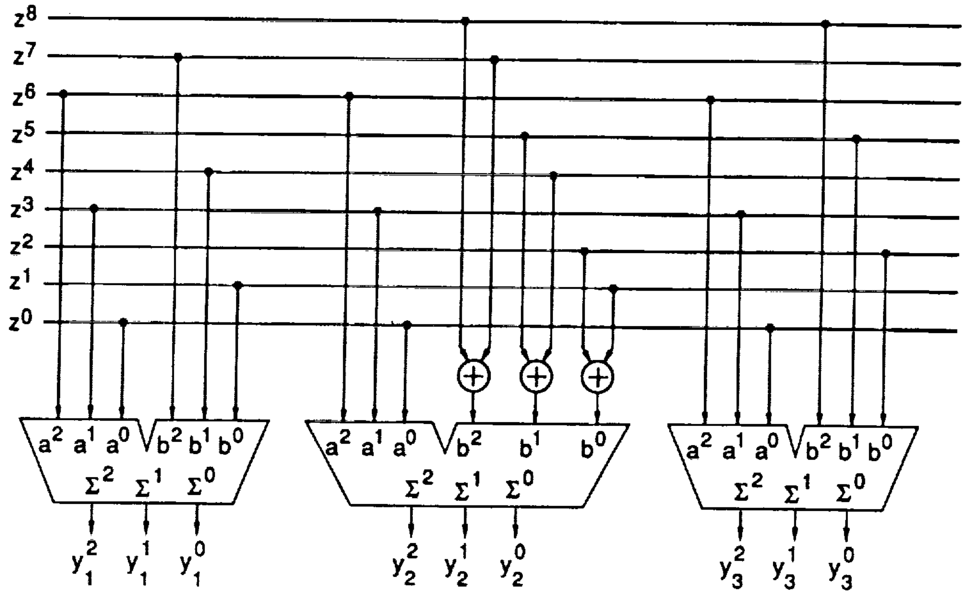

z^5
z^4
z^3
z^2
z^1
z^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
mod-8 adder
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0

(b)

$$\begin{tikzcd}
 & & \Omega^2(0) = \Omega^2 \arrow[r] \arrow[dr] & \Omega^3(0) = \Omega^3 \\
 & \Omega^1(0) = \Omega^1 \arrow[ur] \arrow[dr] & & \Omega^3(4) = \Omega^3 + \begin{bmatrix} 0 \\ 2 \end{bmatrix} \\
 & & \Omega^2(2) = \Omega^2 + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \arrow[r] \arrow[dr] & \Omega^3(2) = \Omega^3 + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
\Omega^0(0) = \Omega^0 \arrow[uur] \arrow[ddr] & & & \Omega^3(6) = \Omega^3 + \begin{bmatrix} 1 \\ 3 \end{bmatrix} \\
 & & \Omega^2(1) = \Omega^2 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \arrow[r] \arrow[dr] & \Omega^3(1) = \Omega^3 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
 & \Omega^1(1) = \Omega^1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \arrow[ur] \arrow[dr] & & \Omega^3(5) = \Omega^3 + \begin{bmatrix} 0 \\ 3 \end{bmatrix} \\
 & & \Omega^2(3) = \Omega^2 + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \arrow[r] \arrow[dr] & \Omega^3(3) = \Omega^3 + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \\
 & & & \Omega^3(7) = \Omega^3 + \begin{bmatrix} 1 \\ 4 \end{bmatrix}
\end{tikzcd}$$

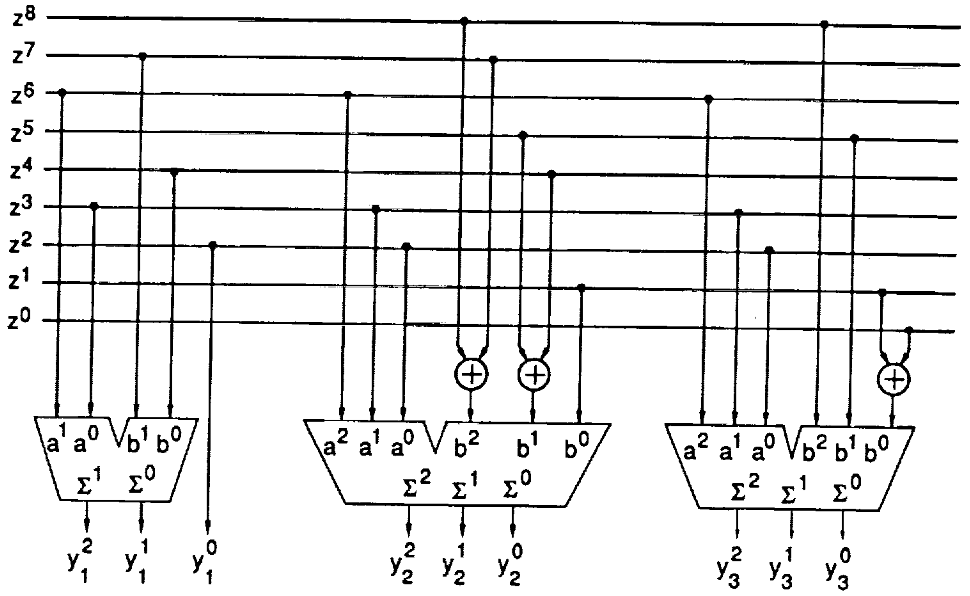
$p = 0$	$p = 1$	$p = 2$	$p = 3$
$\Omega^0 = \Omega(C_0, C_0, C_0)$	$\Omega^1 = \Omega(C_0, C_0, C_1)$	$\Omega^2 = \Omega(C_0, C_0, C_2)$	$\Omega^3 = \Omega(C_0, C_1, C_2)$
$\Delta_0^2 = 0.586$	$\Delta_1^2 = 1.172$	$\Delta_2^2 = 2.0$	$\Delta_3^2 = 4.0$


z
12
T$_1$
z$'_1$
6
z$'_2$
6
T$_2$
T$_2$
3
3
3
3
y$_1$
y$_2$
y$_3$
y$_4$



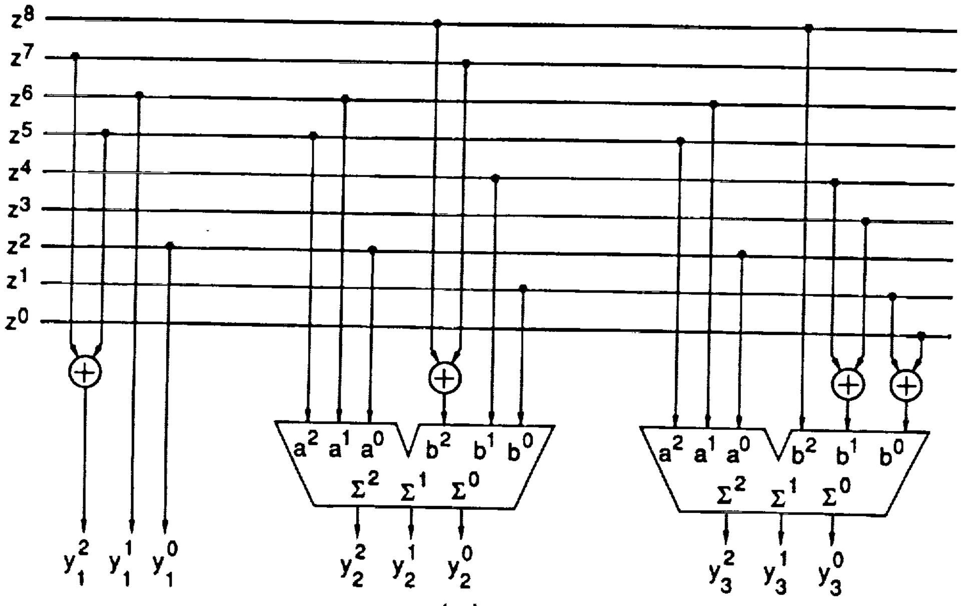
z^8 z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0
y_3^2 y_3^1 y_3^0

(a)



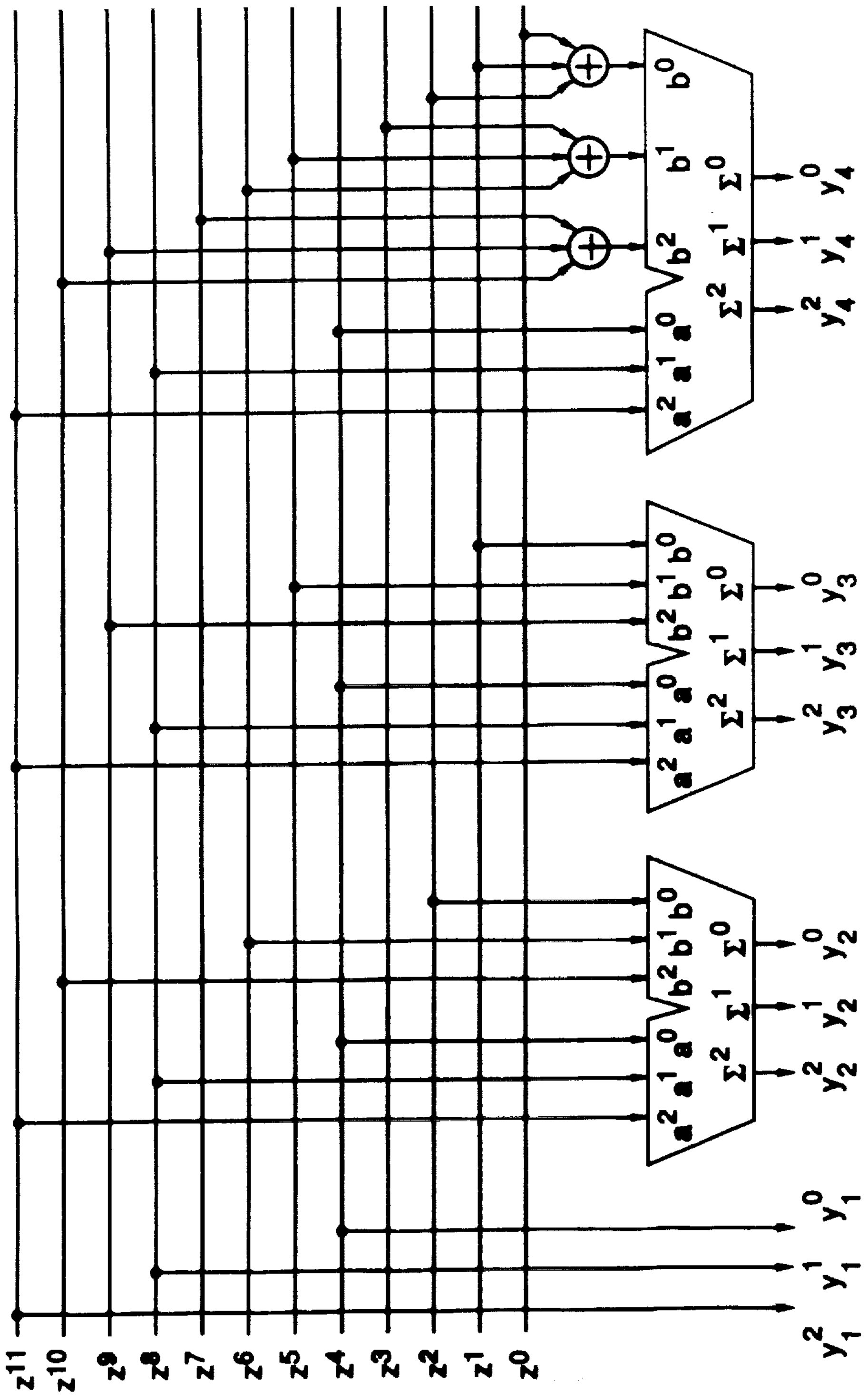
z^8 z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0
a^1 a^0 b^1 b^0
Σ^1 Σ^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0
y_3^2 y_3^1 y_3^0

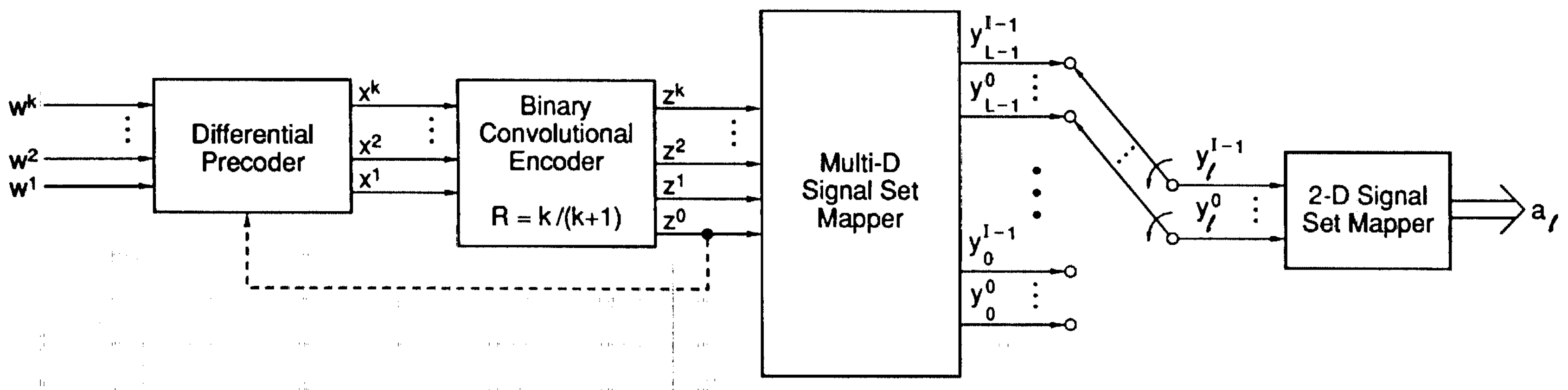
(b)

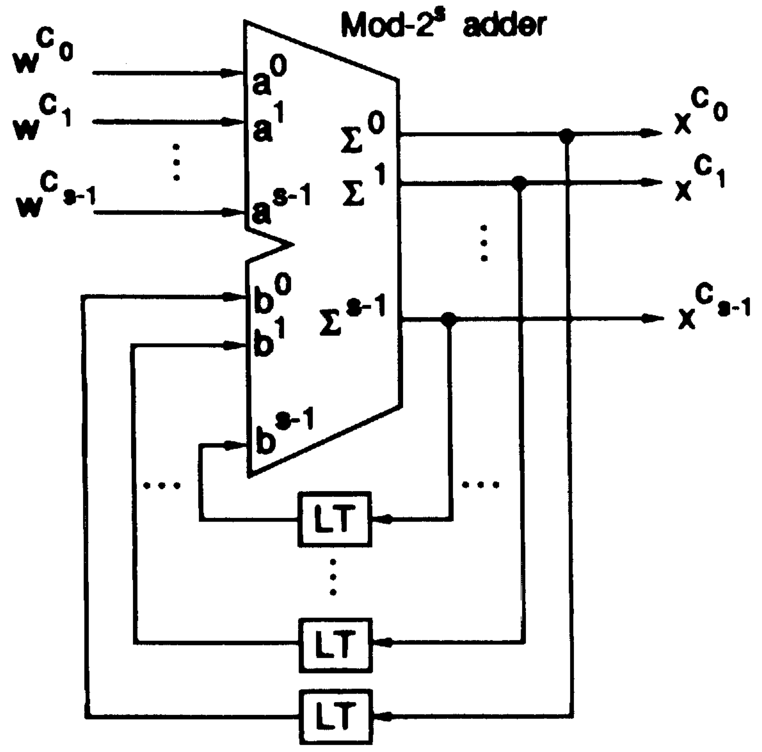


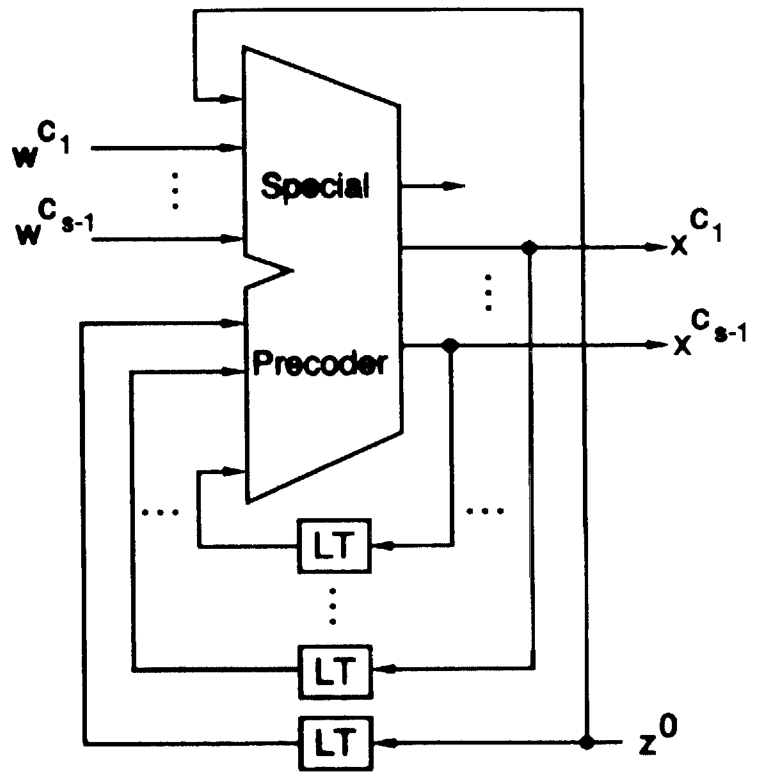
z^8 z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0
y_3^2 y_3^1 y_3^0

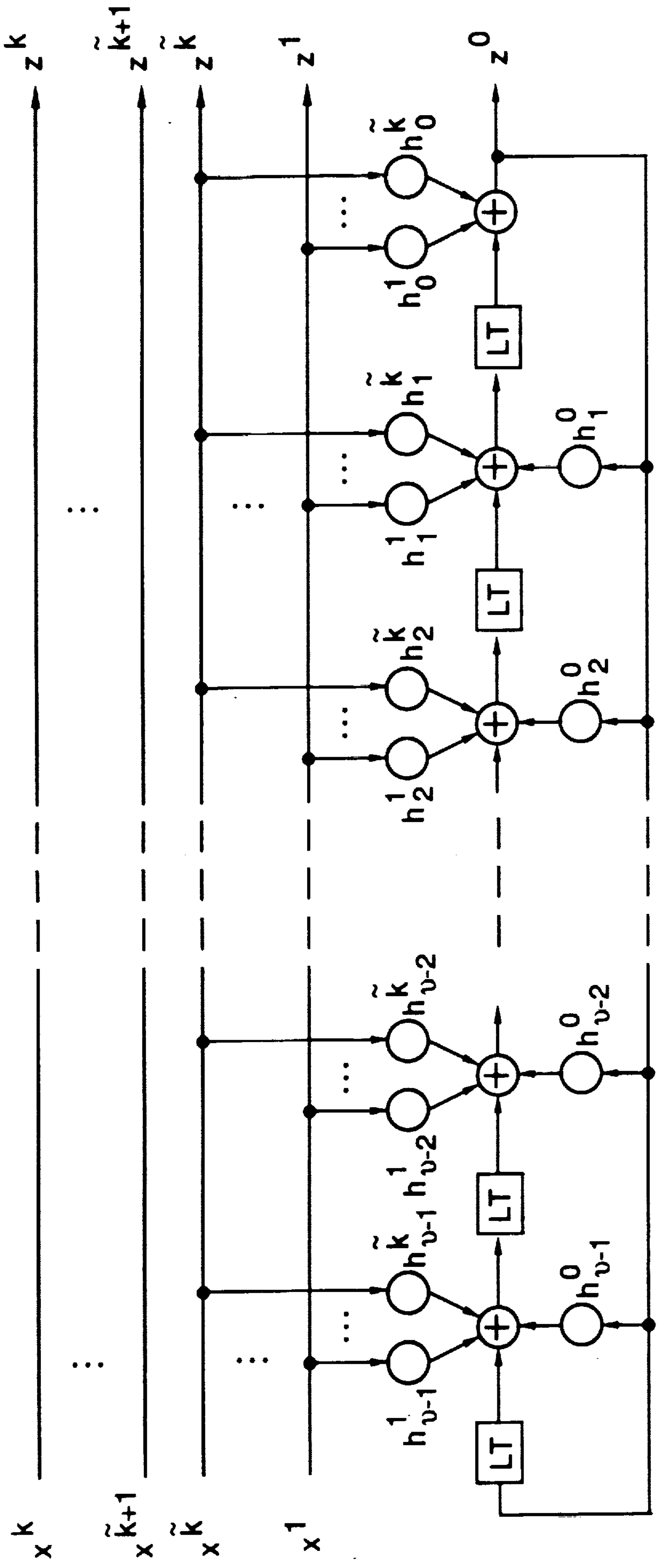
(c)


z^{11} z^{10} z^9 z^8 z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0
y_1^2 y_1^1 y_1^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_2^2 y_2^1 y_2^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_3^2 y_3^1 y_3^0
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_4^2 y_4^1 y_4^0

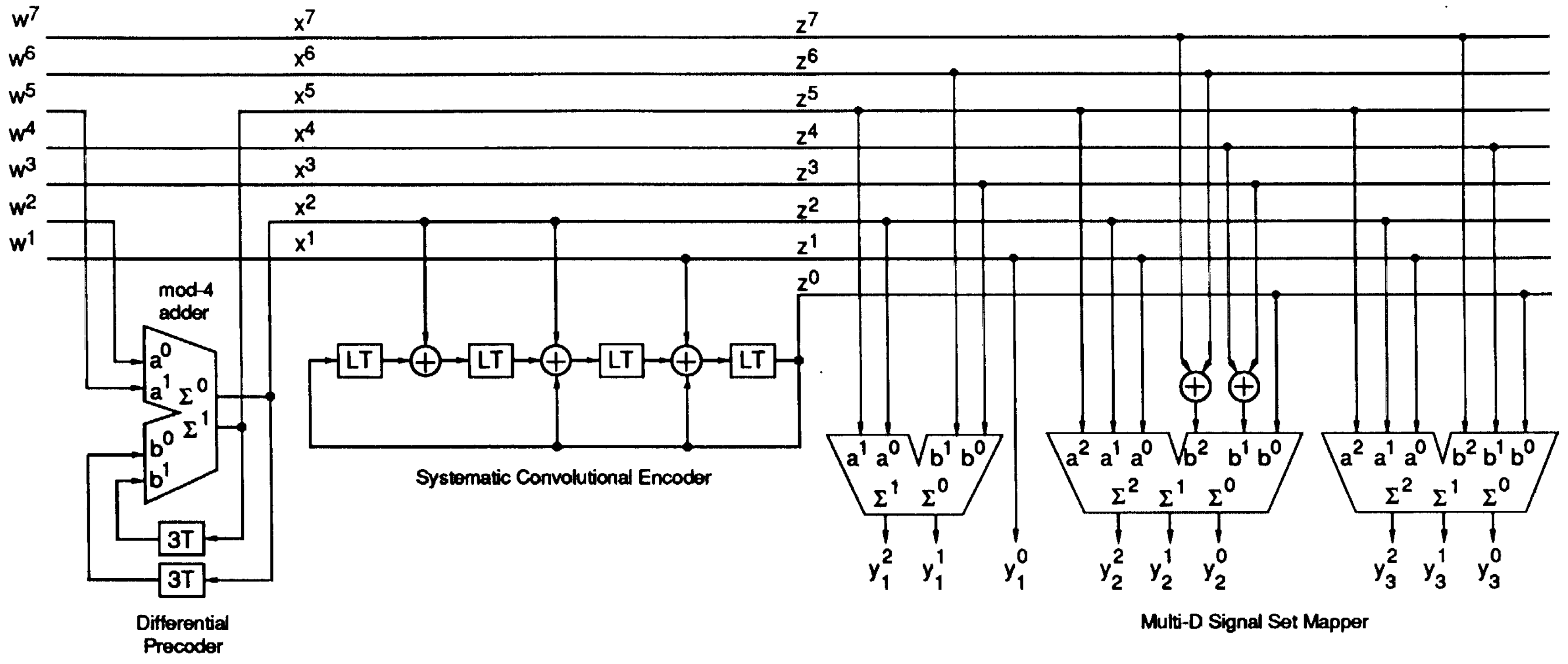

w^k
w^2
w^1
Differential Precoder
x^k
x^2
x^1
Binary Convolutional Encoder
R = k/(k+1)
z^k
z^2
z^1
z^0
Multi-D Signal Set Mapper
y^{I-1}_{L-1}
y^{0}_{L-1}
y^{I-1}_{0}
y^{0}_{0}
y^{I-1}_{l}
y^{0}_{l}
2-D Signal Set Mapper
a_l


Mod-2^s adder
w^{C_0}
w^{C_1}
$w^{C_{s-1}}$
a^0
a^1
a^{s-1}
b^0
b^1
b^{s-1}
Σ^0
Σ^1
Σ^{s-1}
x^{C_0}
x^{C_1}
$x^{C_{s-1}}$
LT
LT
LT
(a) $C_0 > 0$

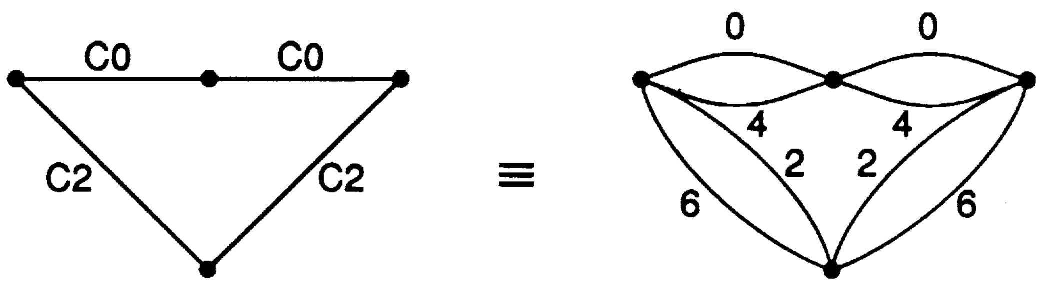

w^{C_1}
$w^{C_{s-1}}$
Special
Precoder
x^{C_1}
$x^{C_{s-1}}$
z^0
LT
LT
LT
(b) $C_0 = 0$

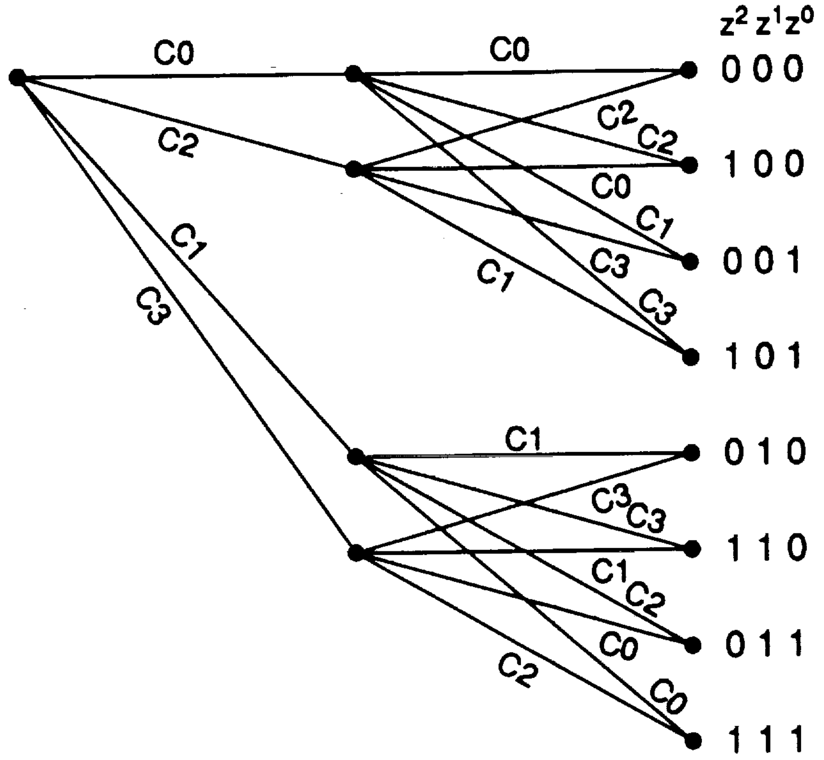


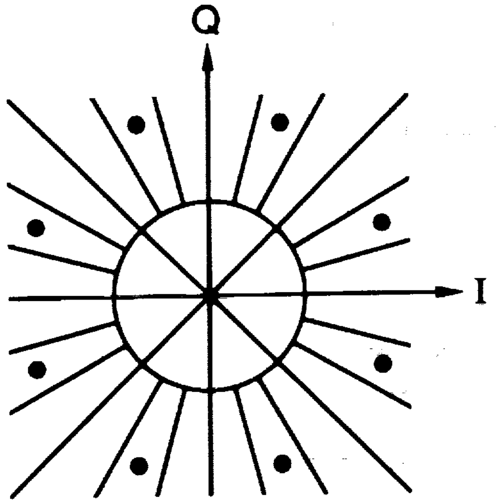
x^k
$x^{\tilde{k}+1}$
$x^{\tilde{k}}$
x^1
z^k
$z^{\tilde{k}+1}$
$z^{\tilde{k}}$
z^1
z^0
LT
$h^1_{\upsilon-1}$
$h^{\tilde{k}}_{\upsilon-1}$
$h^0_{\upsilon-1}$
$h^1_{\upsilon-2}$
$h^{\tilde{k}}_{\upsilon-2}$
$h^0_{\upsilon-2}$
h^1_2
$h^{\tilde{k}}_2$
h^0_2
h^1_1
$h^{\tilde{k}}_1$
h^0_1
h^1_0
$h^{\tilde{k}}_0$

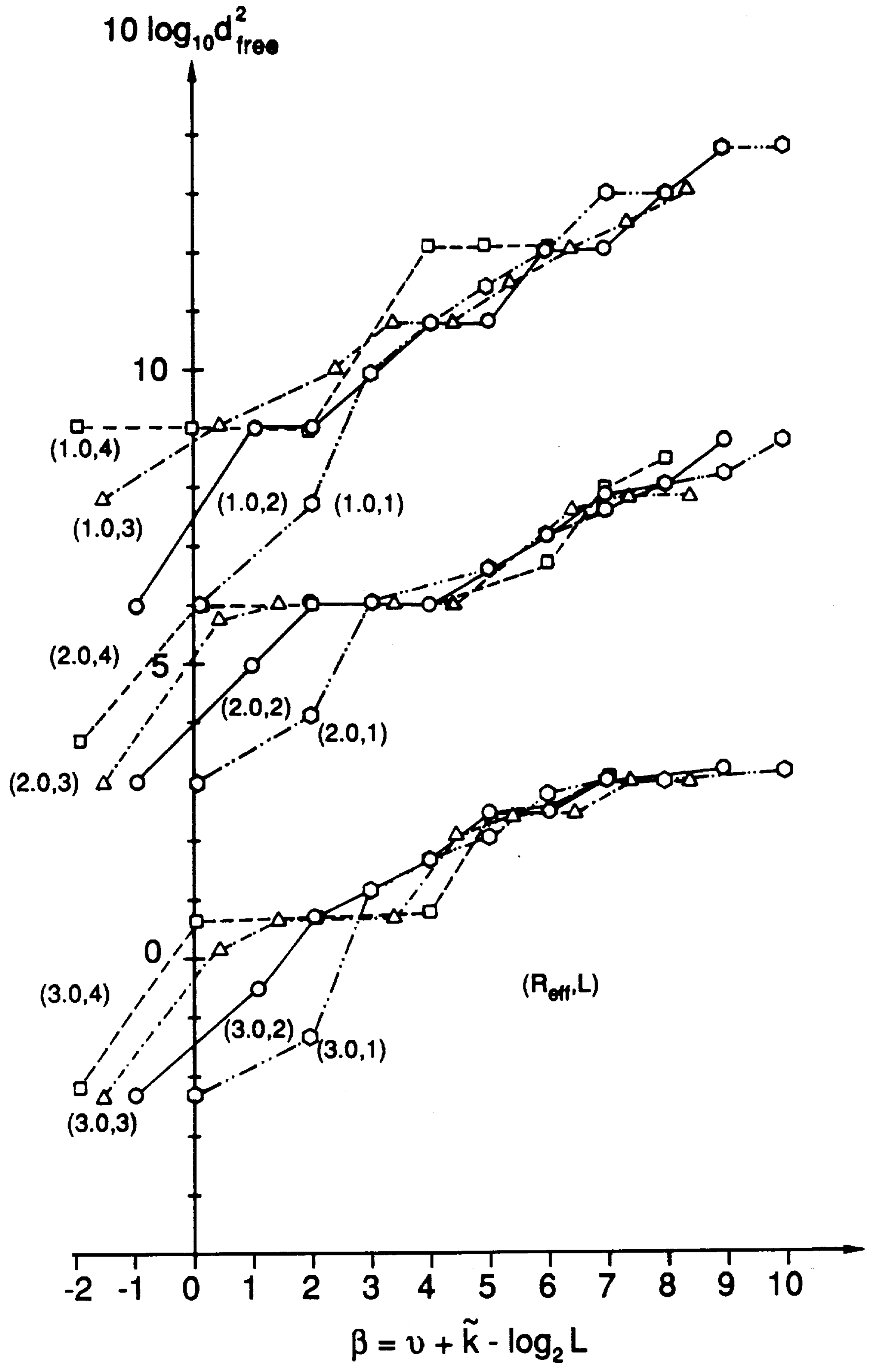


w7
w6
w5
w4
w3
w2
w1
x7
x6
x5
x4
x3
x2
x1
z7
z6
z5
z4
z3
z2
z1
z0
mod-4 adder
a0 a1 Σ0 Σ1 b0 b1
3T
3T
Differential Precoder
LT
LT
LT
LT
Systematic Convolutional Encoder
a1 a0 b1 b0 Σ1 Σ0
a2 a1 a0 b2 b1 b0 Σ2 Σ1 Σ0
a2 a1 a0 b2 b1 b0 Σ2 Σ1 Σ0
y1 2, y1 1, y1 0
y2 2, y2 1, y2 0
y3 2, y3 1, y3 0
Multi-D Signal Set Mapper

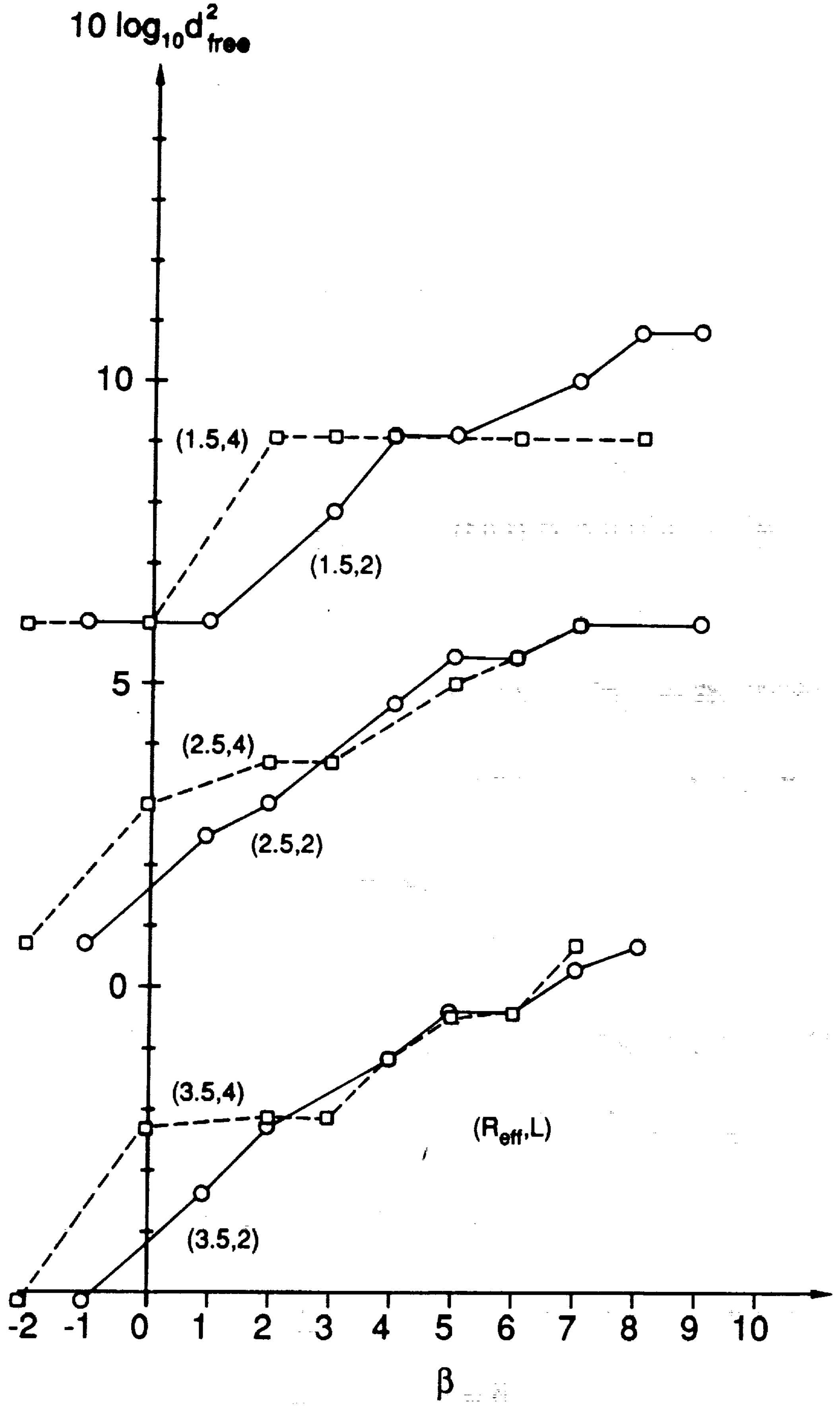

C0
C0
C2
C2
≡
0
0
4
4
2
2
6
6


z² z¹ z⁰
C0
C2
C1
C3
C0
C2
C2
C0
C1
C1
C3
C3
C1
C3
C3
C1
C2
C0
C2
C0
000
100
001
101
010
110
011
111

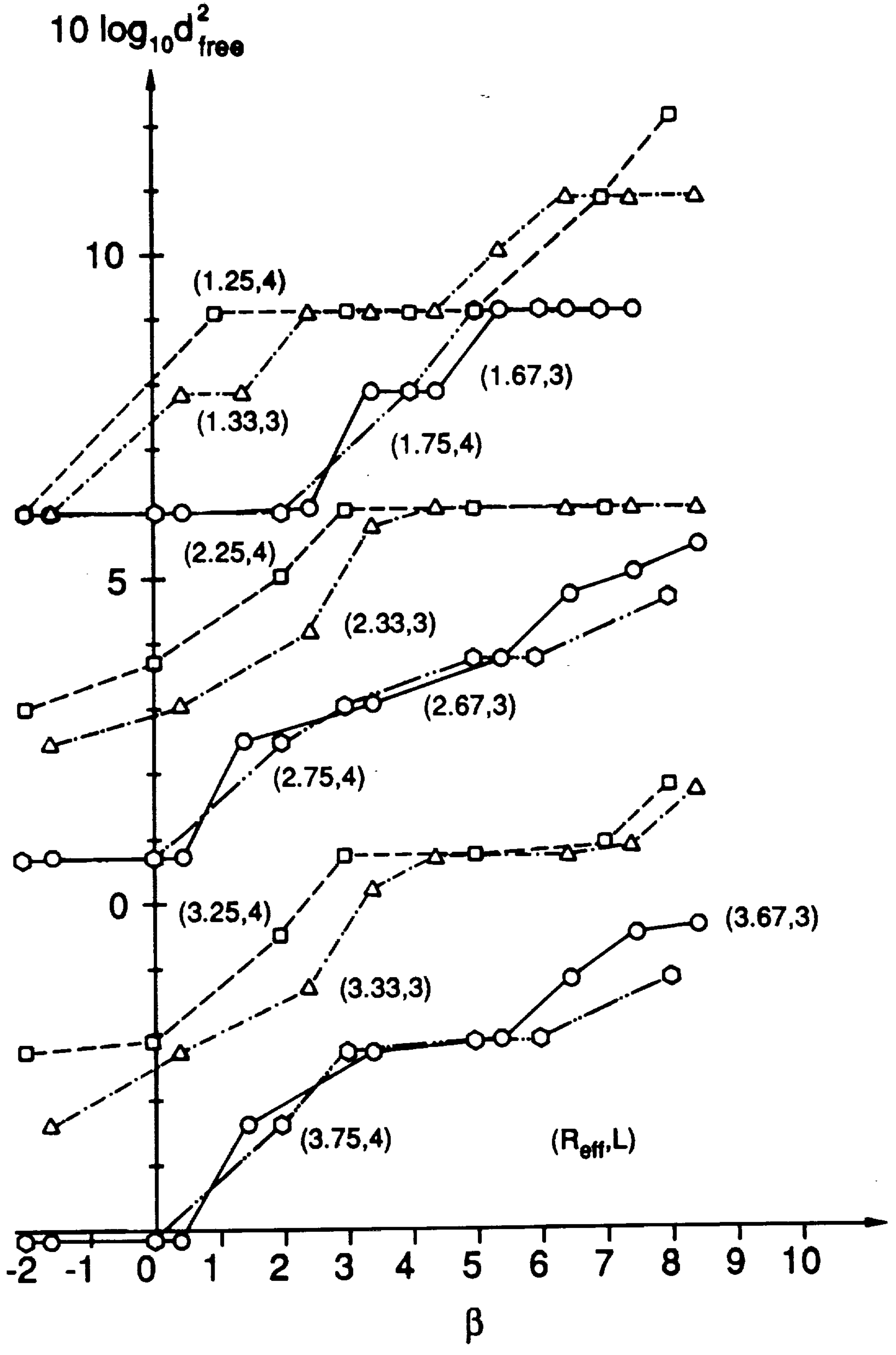

Q
I



$10 \log_{10} d^2_{free}$
10
5
0
(1.0,4)
(1.0,3)
(1.0,2)
(1.0,1)
(2.0,4)
(2.0,3)
(2.0,2)
(2.0,1)
(3.0,4)
(3.0,3)
(3.0,2)
(3.0,1)
(R_{eff}, L)
-2 -1 0 1 2 3 4 5 6 7 8 9 10
$\beta = \upsilon + \tilde{k} - \log_2 L$



$10 \log_{10} d^2_{free}$
10
5
0
(1.5,4)
(1.5,2)
(2.5,4)
(2.5,2)
(3.5,4)
(3.5,2)
(R_{eff}, L)
-2
-1
0
1
2
3
4
5
6
7
8
9
10
β


10 $\log_{10} d^2_{free}$
10
5
0
-2 -1 0 1 2 3 4 5 6 7 8 9 10
β
(1.25,4)
(1.33,3)
(1.67,3)
(1.75,4)
(2.25,4)
(2.33,3)
(2.67,3)
(2.75,4)
(3.25,4)
(3.33,3)
(3.67,3)
(3.75,4)
(R_{eff},L)

Appendix B

On the Performance of Multi-Dimensional Phase Modulated Trellis Codes

On the Performance of Multi-Dimensional Phase Modulated Trellis Codes*

Lance C. Perez
Daniel J. Costello, Jr.

Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

October 11, 1989

*This work was supported in part by NASA grants NAG5-557 and NGT-70109.

1 Introduction

Demands for higher information rates (spectral efficiencies) in satellite communications are such that this traditionally power limited channel is now considered to be bandlimited as well. This has made high speed satellite channels a natural candidate for the application of Trellis Coded Modulation (TCM). Introduced by Ungerboeck [1],[2], TCM achieves significant asymptotic coding gains relative to an uncoded system of the same information rate with no bandwidth expansion. Due to the extensive use of nonlinear amplifiers, such as traveling wave tubes, in satellite communications systems only TCM schemes using MPSK signal constellations are appropriate. A class of multidimensional (multi-D) MPSK trellis codes has been proposed for use in NASA high speed satellite communication links [3]. These codes have greater flexibility in achievable information rate, better rotational invariance properties, and higher speed decoder implementations than regular two dimensional (2-D) Ungerboeck codes.

A typical satellite communication system consists of a Reed-Solomon(RS) outer code, a symbol interleaver, and an inner code (Figure 1). The satellite channel is usually well modeled by the zero mean Additive White Gaussian Noise (AWGN) process with two sided power spectral density $N_o/2$. In one mode of operation, the RS code and interleaver are bypassed and the information bits are directly encoded by the inner code and transmitted. In this case, a bit error rate (BER) of $\sim 10^{-5}$ out of the inner decoder is considered reasonable. Under poor channel conditions or when extremely low error rates are required, the entire concatenated coding system is used. For a given (N, K) RS code over $GF(2^b)$ and assuming ideal interleaving, the performance of the system is determined by the symbol error rate (SER) out of the inner decoder, where the symbol size, b, is determined by the RS code. Thus, the choice of the inner code is critical when designing such a system.

In this paper, the performance of multi-D MPSK trellis codes as the inner code in a satellite communication system is studied through extensive computer simulation. In Section 2, a brief review of the results in [3] is presented in order to introduce multi-D MPSK signal sets and the multi-D encoder structure. In Section 3, the BER performance of multi-D MPSK trellis codes on the AWGN channel with soft decision Viterbi decoding is studied. This section includes an example of the calculation of the Zehavi and Wolf transfer function bound and results on the minimum truncation length required for Viterbi decoding of MPSK trellis codes. Finally, the performance of MPSK trellis codes in a concatenated coding system is studied in Section 4 through the use of computer simulation to obtain the SER out of the inner decoder.

2 Multi-Dimensional Trellis Codes

The fundamental concept of 2-D TCM is that by trellis coding onto an expanded signal set relative to that needed for uncoded transmission, a larger free distance and an associated improvement in performance may be achieved without bandwidth expansion or a reduction in information rate. In [1] and [2], Ungerboeck considered the 2-D MPSK case where uncoded 2^m PSK was expanded into a rate $m/(m+1)$, 2^{m+1} PSK trellis code. In multi-D MPSK TCM, the same principal is applied to a signal constellation, denoted LxMPSK, constructed from the Cartesian product of L 2D-MPSK signal sets. Each branch of the trellis is then labelled with an ordered sequence of L 2-D signals and a 2L-dimensional signal is realized by transmitting a sequence of L 2-D signals. The efficient labelling of trellis branches is accomplished through a systematic binary set partitioning technique based on block codes and their cosets.

The class of LxMPSK codes studied here has the same schematic encoder diagram introduced by Forney et.al.[4] for Ungerboeck codes with one slight modification. As shown in Figure 2, the multi-D encoder allows the q least significant bits to be fixed, nominally to 0, during each coding interval. Thus, whereas 2D-MPSK TCM allowed only rate $m/m+1$ codes with information rates (R) of 1.0, 2.0, and 3.0 bits per modulation interval(T) for QPSK, 8PSK, and 16PSK, respectively, multi-D MPSK allows the construction of rate $(Lx\log_2(M-q-1))/(Lx\log_2 M)$ codes with a variety of information rates ranging from 1.0 bit/T to 3.75 bits/T.

As an example, consider the familiar 4-state Ungerboeck 8PSK code shown in Figure 3 with information rate 2.0 bits/T. The fundamental trellis structure of this code is determined by the rate 1/2 convolutional encoder and is shown in Figure 4. During each coding interval, the trellis encoder receives $m = 2$ information bits, of which $\tilde{m} = 1$ are encoded by the rate 1/2 convolutional encoder, and maps them to a single 8PSK signal. This results in the trellis structure of Figure 4 with each transition replaced by

$2^{m-\tilde{m}} = 2^1$ parallel transitions. The same rate 1/2 convolutional encoder can be used to construct a 2x8PSK code with information rate 2.5 bits/T. In this case, the trellis encoder receives $m = 5$ information bits per coding interval, of which the least significant bit is encoded by the convolutional encoder. The $m + 1 = 6$ bits are then mapped to a single 2x8PSK signal. This results in a rate 5/6 code with the trellis structure of Figure 4 where each branch is replaced with $2^{m-\tilde{m}} = 2^4 = 16$ parallel transitions and each transition is labelled with a 2x8PSK signal. If the first $q = 1$ bits into the trellis encoder are now set to 0, a rate 4/6, 2x8PSK code with information rate 2 bits/T is obtained. This code has the trellis structure of Figure 4 with $2^{4-1} = 8$ parallel transitions and each transition is labelled with a 2x8PSK signal from the left branch of the first level of the 2x8PSK binary partition (see Pietrobon [3] et. al. for a discussion of multi-D signal set partitioning). It should be noted that, though this example illustrates the concept of these LxMPSK trellis codes, the best LxMPSK codes will in general not have the same convolutional encoder as a 2-D Ungerboeck code.

3 Code Performance and Simulation Results

Having introduced the basic concepts of LxMPSK trellis codes, it is now of interest to investigate their performance. It is common practice to measure the performance of trellis codes in terms of the asymptotic coding gain relative to an uncoded system with the same rate. In this case, the coding gain is given by

$$\gamma = 10 \log_{10} \left[\frac{d^2_{free_{TCM}}}{d^2_{free_{uncoded}}} \right] \tag{1}$$

The asymptotic coding gain, however, is accurate only as the SNR becomes very large and does not provide much insight into a codes real coding gain at a particular SNR or error rate. In order to determine real coding gains, analytical bounds or simulations must be used.

For linear convolutional codes, the most common analytical method for determining code performance is the transfer function bound. This method uses a modified state diagram to enumerate the complete set of possible error paths from the correct sequence. Inherent in this technique is the assumption that considering the all zero sequence as the correct sequence entails no loss of generality. For most TCM schemes, the mapping from code sequences to channel symbols is non-linear and the set of error paths is dependent on which code sequence was sent. Thus, it is necessary to enumerate the set of all error paths for each possible correct code sequence. This requires a state diagram using $2^{2\nu}$ states and quickly becomes impractical[5]. Zehavi and Wolf [6] have shown that for a large class of trellis codes, including those discussed in [1],[2], and [3], a transfer function upper bound on the average probability of error can be calculated using a modified state diagram having only 2^{ν} states. As with convolutional codes, this procedure becomes intractable as the number of state increases. For multi-D trellis codes, the Zehavi and Wolf method becomes even more complex due to an exponential increase in the number of error vectors whose distance profile from each possible code vector must be enumerated. The calculation of the Zehavi and Wolf bound and its computational difficulties is best manifest through an example.

Consider, the 8PSK, $\nu = 2$, Ungerboeck code introduced in Section 2 and shown in Figure 3. A necessary condition for the application of this bound is that both subsets at the first level of the binary partition tree must have the same distance profile with respect to a binary error vector E. In this case, it is simple to check that the two subsets A=$\{S_0, S_2, S_4, S_6\}$ and Â=$\{S_1, S_3, S_5, S_7\}$ have the same profile for all 8 possible three bit error vectors. For example, the error vector 011 has squared Eulcidean distances $\{3.4142, 0.5858, 3.4142, 0.5858\}$ from A and $\{0.5858, 3.4142, 0.5858, 3.4142\}$ from Â. With this condition, the distance profile for each of the 8 error vectors with respect to either subset can be determined as shown in Table 1. Note, that for a LxMPSK code this involves finding the distance profiles of $2^{L\log_2(M)}$ error vectors with respect to $M^L/2$ signal points. From Table 1, the distance polynomial of each error vector can be obtained as in Table 2, where the power of W is the squared Euclidean distance and the coefficient is the number of error paths with that distance. This information is used to construct a modified state diagram that enumerates the multiplicity and distance of all possible error paths.

The state diagram of the R=1/2 convolutional code used in this trellis code is shown in Figure 5. The diagram of Figure 5 is transformed into modified state diagram that enumerates all possible error paths by labelling each branch with a polynomial consisting of terms of the form $L^i I^j W^\alpha$ where i is the number of symbols, j is the number of nonzero information bits, and α is the squared Euclidean distance. The key to labelling the branches of the modified state diagram is to remember that each transition in the convolutional encoder's state diagram represents a group of parallel transtions for the trellis code. For example, when considering the transiton from state 1 to state 3 in Figure 5 the polynomials of the error vectors 011 and 111 must be taken into account yielding the polynomial $2LI(2W^{0.58579} + 2W^{3.4142}) + 2LI^2(2W^{0.58579} + 2W^{3.4142})$ as the appropriate label. Repeating this procedure for each branch gives the modified state diagram of Figure 6.

Regarding this as a signal flow graph, the transfer function is found to be

$$T(W, L, I) = 4LIW^4 + \frac{L^4I^4\alpha^2\beta^2 + L^3I^2\alpha^2\gamma(1 - L\gamma)}{1 - L^3I^2\beta^2\delta - L^2\gamma\delta - L\gamma + L^3\gamma^2\delta} \tag{2}$$

where $\alpha = 4(I + 1)W^2$, $\beta = 2(I + 1)(W^{0.58579} + W^{3.4142})$, $\gamma = 4(W^{0.58579} + IW^{3.4142})$, and $\delta = 4(1 + IW^4)$. Using this transfer function, Zehavi and Wolf show that the bit error rate for a trellis code with maximum likelihood decoding (MLD)is upper bounded by

$$P_{b_{MLD}} \leq \frac{1}{n-1} Q\left(\sqrt{\frac{d_f^2 E_s}{2N_0}}\right) \exp\left(\frac{d_f^2 E_s}{4N_0}\right) \frac{\partial T(W, L, I)}{\partial I}\Bigg|_{W=\exp(-E_s/4N_0), L=2^{1-n}, I=1} \tag{3}$$

where d_f^2 is the normalized squared free Euclidean distance of the code. The Zehavi and Wolf bound and corresponding simulation results are plotted in Figures 7 and 8 for the Ungerboeck code discussed above and a 2x8PSK, R=2.5 bits/T, 4-D code from [3], respectively. As expected, the bound and the simulation results conform closely except at low SNR's where the union bound becomes relatively loose. This of practical interest since many satellite communications systems operate at low SNR's and concatenated systems are sensitive to small deviations in the performance of the inner code.

Simulation results for these codes were obtained via a stochastic or Monte Carlo simulation of a soft decision MLD Viterbi decoder. This involves simulating the AWGN channel with a computer generated pseudo-random number generator that approximates the Gaussian distribution. The Marsaglia-Bray[7] method was used to implement the Gaussian random number generator from the available uniform random number generators. Because trellis codes are non-linear, a number of nonzero information sequences are considered in each simulation and the code performance is determined as the average over these sequences.

Figure 9 shows the simulation results for a group of $\nu = 2$, Lx8PSK trellis codes with information rate 2.0 bits/T and $d^2_{free} = 4.0$. These codes show a coding gain from 2.0dB to 2.4dB at $\sim 10^{-5}$ compared to uncoded QPSK. The 4, 6, and 8 dimensional codes suffer a 0.2dB to 0.4dB loss in performance compared to the Ungerboeck 8PSK code due to higher path multiplicities and denser distance spectra. For example, the 4x8PSK code has a multiplicity $N_{free} = 12$ of the minimum free distance path ($d^2_{free} = 4.0$) and a second spectral line at $d^2_2 = 4.686$ with multiplicity $N_2 = 32$. On the other hand, it is possible to implement a 4x8PSK decoder at 4 times the speed of the Ungerboeck code by using read only memories (ROM's) to decode the parallel transitions. Analagous simulation results were obtained for the multi-D Lx4PSK codes with information rate 1 bit/T and the Lx16PSK codes with information rate 3 bits/T. Figure 10 shows simulation results for $\nu = 4$ and $\nu = 6$, $R = 2.67$ bits/T, 3x8PSK codes compared to a $\nu = 4$ 8PSK periodically time varying trellis code (PTVTC) [8] of the same effective rate. The 3x8PSK trellis codes show an improvement of 0.5 dB for $\nu = 4$ and 1.0 dB for $\nu = 6$ and are capable of a decoder implementation that would operate at three times the speed of the PTVTC decoder. Simulation results for a number of the codes given in [3] are contained in Appendix I .

The ideal Viterbi decoder retains the surviving path into each encoder state and waits until the end of the current code sequence before decoding any information bits. In actual implementations, finite memory limitations allow the decoder to retain only the past τ branches of information bits for each state. The effect of this constraint, called the truncation length of the decoder, is that additional decoding errors are introduced other than those that would occur in an ideal MLD Viterbi decoder. For a particular code, the minimum truncation length is the minimum value of τ for which the performance of the truncated Viterbi decoder will asymptotically approach that of the ideal MLD decoder. In [9] the Zehavi and Wolf bound is modified to account for the truncation length of the Viterbi decoder. Using the modified bound, the performance of a code for a particular τ can be calculated as well as the minimum truncation length required for asymptotically ML performance. In Figure 11, the modified bound and simulation results are plotted for varying τ for the R=2 bits/T, $\nu = 2$, 8PSK Ungerboeck code. Figure 12 shows the truncation length simulation results for the 3x8PSK, R=2.33 bits/T, $\nu = 4$ code currently being implemented. In [9], the minimum truncation length of this code was calculated to be 11 branches.

4 Concatenated Coding and LxMPSK Codes

Many satellite communications links use concatenated coding in order to achieve extremely low BER's over noisy channels with reasonable code complexity. The class of Reed-Solomon (RS) block codes are often chosen as the outer code in these systems due to their optimal distance properties, high rates, and the availability of relatively simple decoding algorithms. Additionally, an interleaver is usually placed between the inner and outer code to ensure that errors in the RS symbols are independent. For a concatenated coding system using a t-error-correcting (N, K) RS outer code and ideal interleaving, the overall bit error rate is well approximated by

$$p_b = \frac{2t+1}{N} \sum_{i=t+1}^{N} \binom{N}{i} p_s^i (1-p_s)^{N-i} \tag{4}$$

where p_s is the RS symbol error rate out of the inner decoder. Computer simulation is necessary to get an accurate measure of p_s for a given inner code.

When using convolutional inner codes, it was shown in [10] that choosing an RS code over $GF(2^b)$ matched to a byte oriented inner convolutional codes of rate k/b resulted in an improvement in the overall sytem BER by nearly an order of magnitude compared to unmatched inner codes of the same rate. Costello and Deng[11] have suggested that when using multi-D LxMPSK inner codes and RS outer codes similar results might apply. In Figure 13, the BER and symbol error rate (SER) for a $\nu = 2$, $R = 2.0$ bits/T, 4x8PSK code and the comparable 2-D Ungerboeck code are shown for a symbol size of 8 bits, i.e. the RS code is over $GF(2^8)$. The results indicate that the BER penalty of 0.2dB to 0.4dB that multi-D codes suffer compared to 2D codes, as noted above, disappears when SER is considered. Figure 14 shows the SER simulation results for the 3x8PSK, $\nu = 4$, $R = 2.33$ bits/T code currently being implemented versus an uncoded 3x8PSK system with the same rate. Thus, the multi-D codes appear to be better suited for concatenated coding than the 2-D codes.

5 Conclusion

The performance of a class of multi-D LxMPSK codes was studied through extensive computer simulation. When compared to 2D Ungerboeck codes with the same information rate, the multi-dimensional codes performed slightly worse due to their high path multiplicities and dense spectra. Additionally, they have a higher trellis complexity due to the larger number of parallel transitions. However, these codes have better rotational invariance properties and are capable of achieving much higher decoder speeds. Furthermore, multi-D LxMPSK codes can be constructed using rates not possible with conventional 2D-MPSK codes. It was also shown that the byte oriented structure of the multi-D codes compensated for the slight loss in BER so that a concatenated system with the proper RS outer code would perform equally as well as a 2D code.

6 References

References

[1] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Parts I and II," *IEEE Commun. Mag.*, Volume 25, Number 2, pp. 5–21, February 1987,

[2] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Trans. Inform. Theory*, **IT-28**, pp. 55-67, January 1982.

[3] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck, and D. J. Costello, Jr., "Trellis Coded Multi-Dimensional Phase Modulation, " to appear in *IEEE Trans. Inform. Theory.*

[4] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient Modulation for Band-Limited Channels," *IEEE Jour. S.A.C.*, **SAC-2**, pp. 632–647, September 1984.

[5] E. Biglieri, "High-level modulation and coding for nonlinear satellite channels," *IEEE Trans. Commun.*, **COM–32**, pp. 616–626, May 1984.

[6] E. Zehavi and J. K. Wolf, "On the Performance Evaluation of Trellis Codes," *IEEE Trans. Inform. Theory*, **IT-33**, pp. 196–202, March 1987.

[7] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1982.

[8] J. M. Nowack and M. A. Herro, "The Effect of Viterbi Decoder Path Memory on the Performance of Trellis Coded Modulation " *Proceedings of the 1989 Conference on Information Sciences and Systems*, pp. 735-740, The Johns Hopkins University, Baltimore, Maryland, March 22-24 1989. pp. 12–21, February 1987.

[9] F. Hemmati and R. J. F. Fang, "Low Complexity Coding Methods for High-Data-Rate Channels," *COMSAT Technical Review*, Volume 16, Number 2, pp. 425–447, Fall 1986.

[10] Lin-nan Lee, "Concatenated Coding Systems Emplying a Unit-Memory Convolutional Code and a Byte-Oriented Decoding Algorithm," *IEEE Trans. Commun.*. **COM-25**, pp. 1064–1074, October 1977.

[11] R. H. Deng and D. J. Costello Jr., "High Rate Concatenated Coding Systems Using Multi-Dimensional Bandwidth Efficient Trellis Inner Codes," to appear in *IEEE Trans. Commun.*.

Table 1: Distance profile for the 8–PSK, $\nu = 2$ Ungerboeck code.

$\mathbf{C} = (C^2C^1C^0)$	$\mathbf{E} = (E^2E^1E^0)$	$d^2(M(\mathbf{C}); M(\mathbf{E}+\mathbf{C}))$
000	000	0
010	000	0
100	000	0
110	000	0
000	001	0.58579
010	001	0.58579
100	001	0.58579
110	001	0.58579
000	010	2
010	010	2
100	010	2
110	010	2
000	011	3.4142
010	011	0.58579
100	011	3.4142
110	011	0.58579
000	100	4
010	100	4
100	100	4
110	100	4
000	101	3.4142
010	101	3.4142
100	101	3.4142
110	101	3.4142
000	110	2
010	110	2
100	110	2
110	110	2
000	111	0.58579
010	111	3.4142
100	111	0.58579
110	111	3.4142

Table 2: Branch Weights for the 8–PSK, $\nu = 2$, Ungerboeck code.

$\mathbf{E} = (E^2E^1E^0)$	Weight Profile
000	4
001	$4W^{0.58579}$
010	$4W^2$
011	$2W^{0.58579} + 2W^{3.4142}$
100	$4W^4$
101	$4W^{3.4142}$
110	$4W^2$
111	$2W^{0.58579} + 2W^{3.4142}$

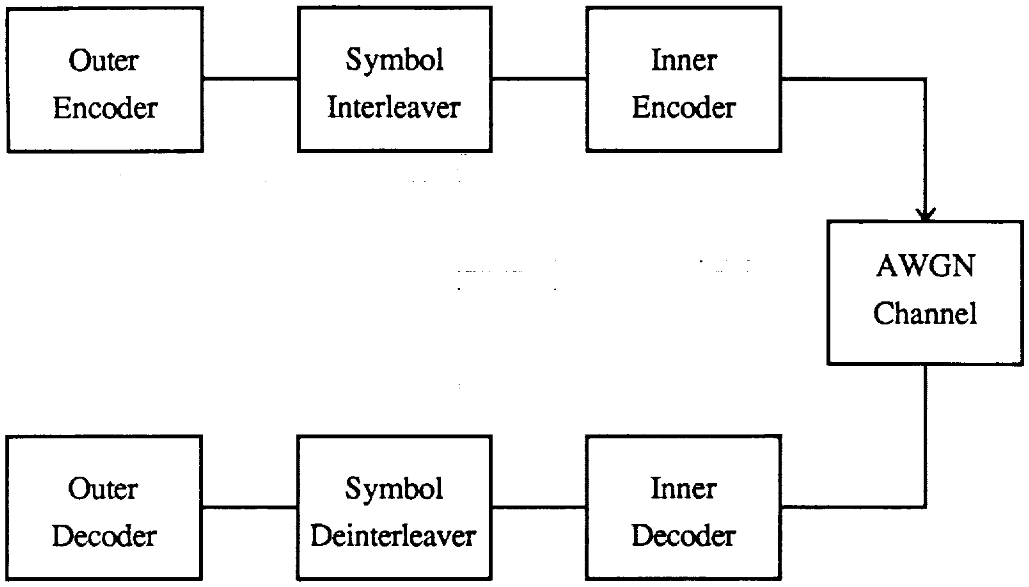


Figure 1: Block Diagram of a Typical Satellite Communications System

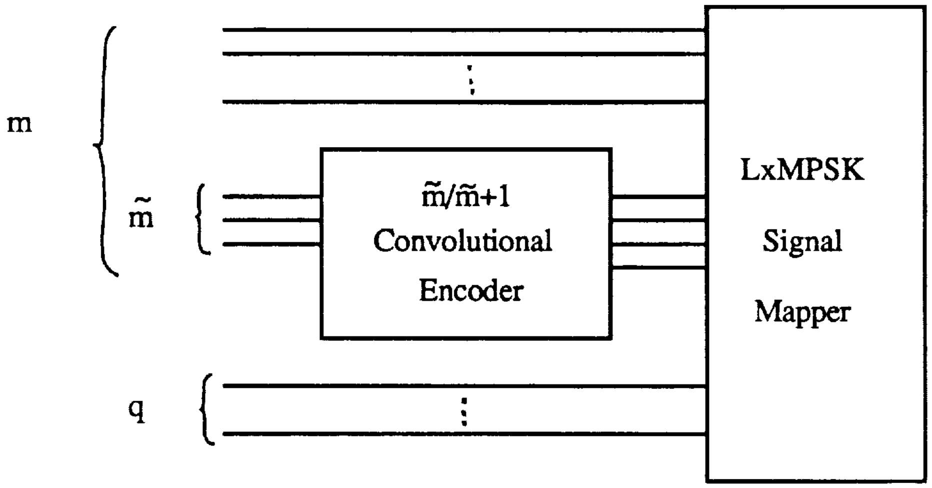


Figure 2: Schematic Representation of a Multi-D Trellis Encoder

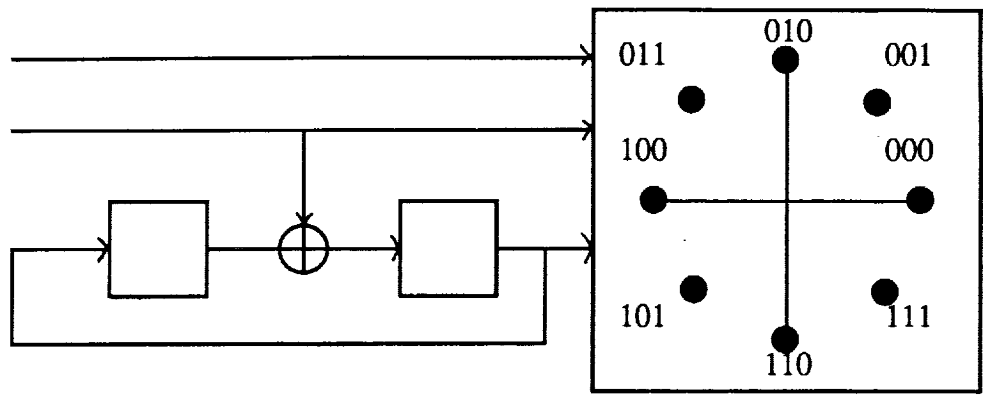


Figure 3: Encoder for the 4 State, 8PSK, R=2/3 Ungerboeck Code

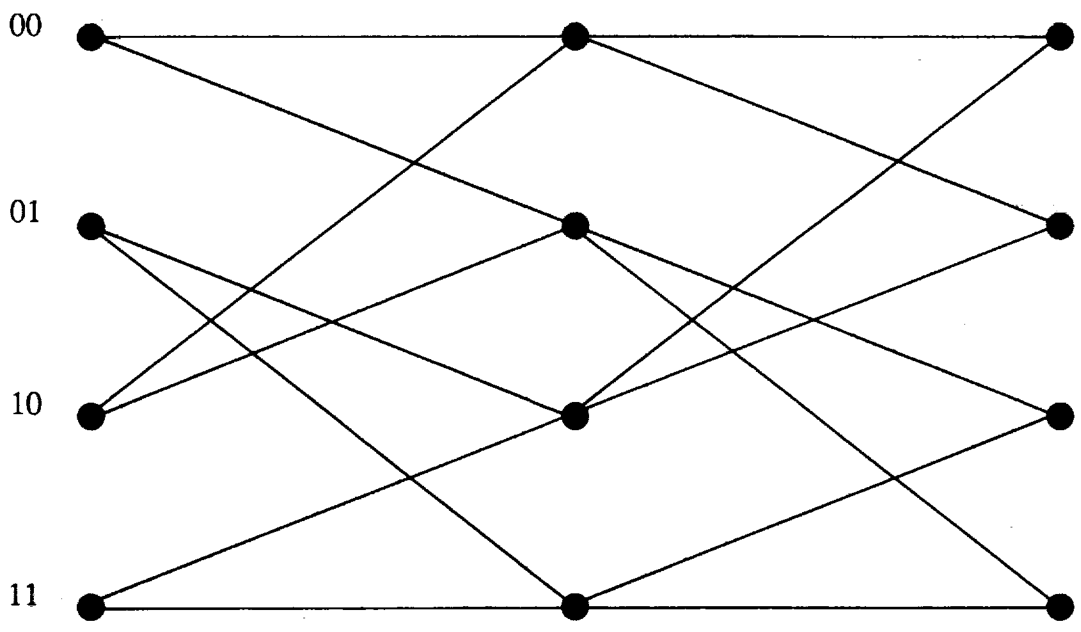


Figure 4: Trellis Diagram for the Rate 1/2 Convolutional Encoder of Figure 3.

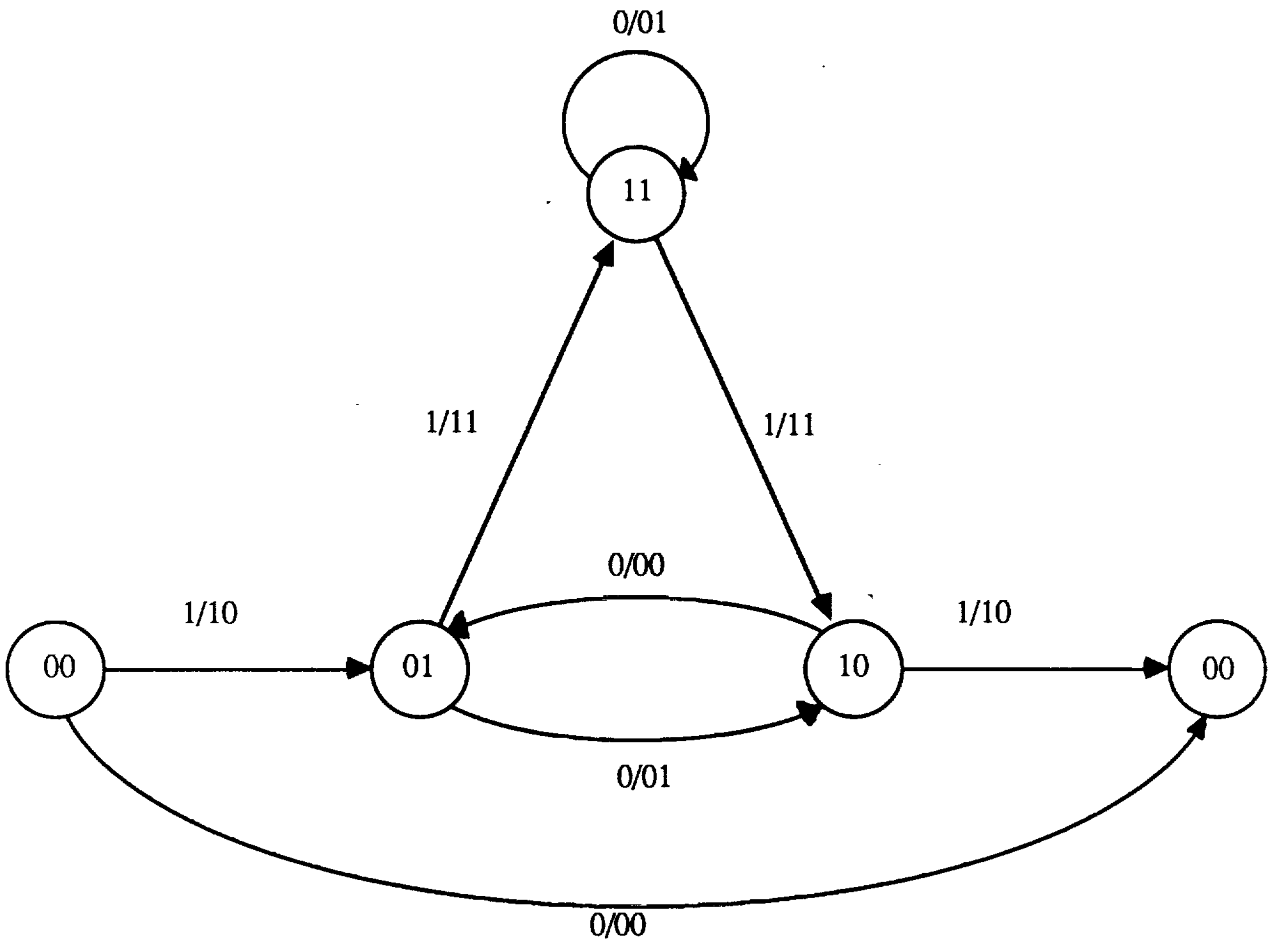


Figure 5: State Diagram for the Convolutional Code of Figure 3.

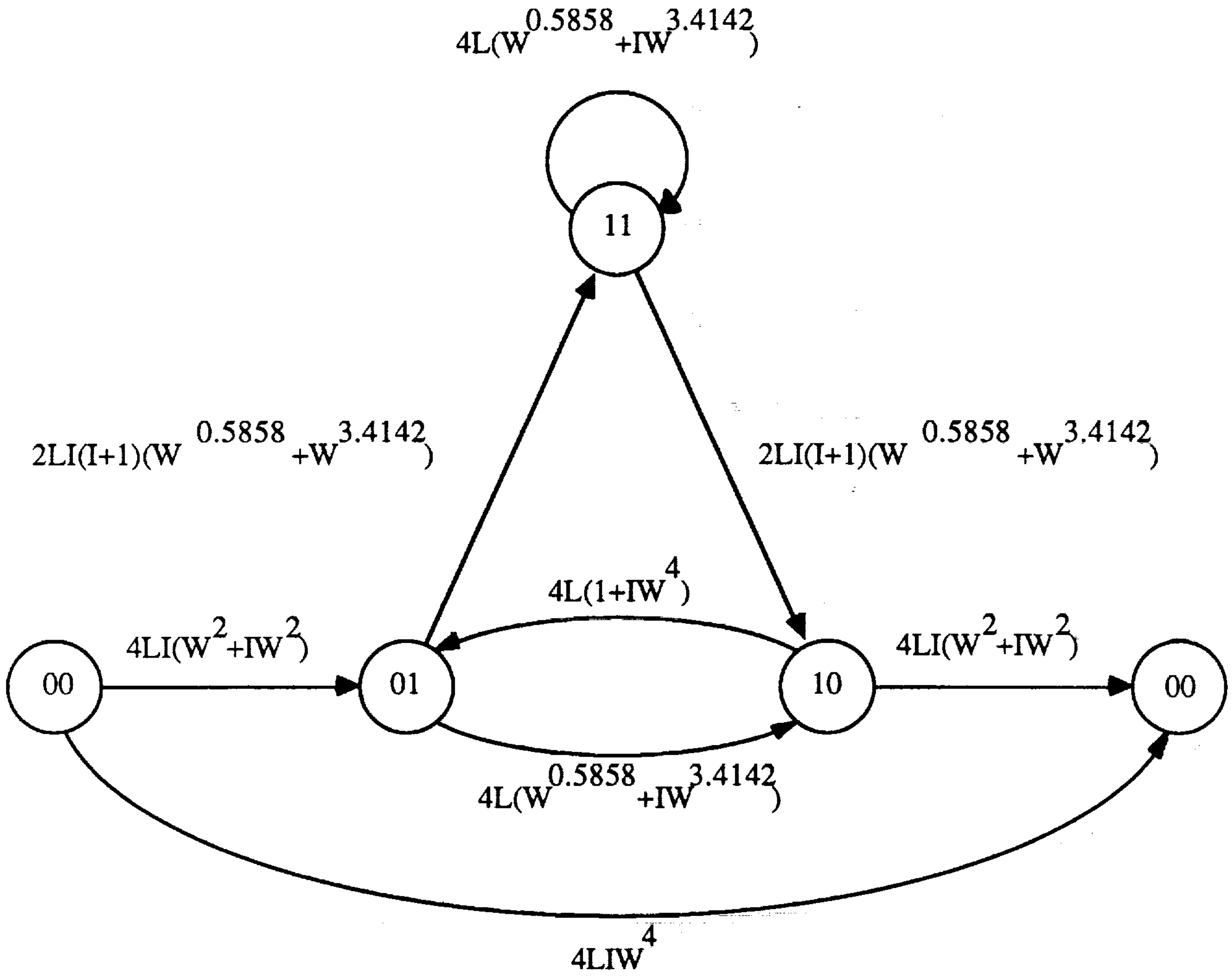


Figure 6: Modified State Diagram for the Trellis Code of Figure 3.

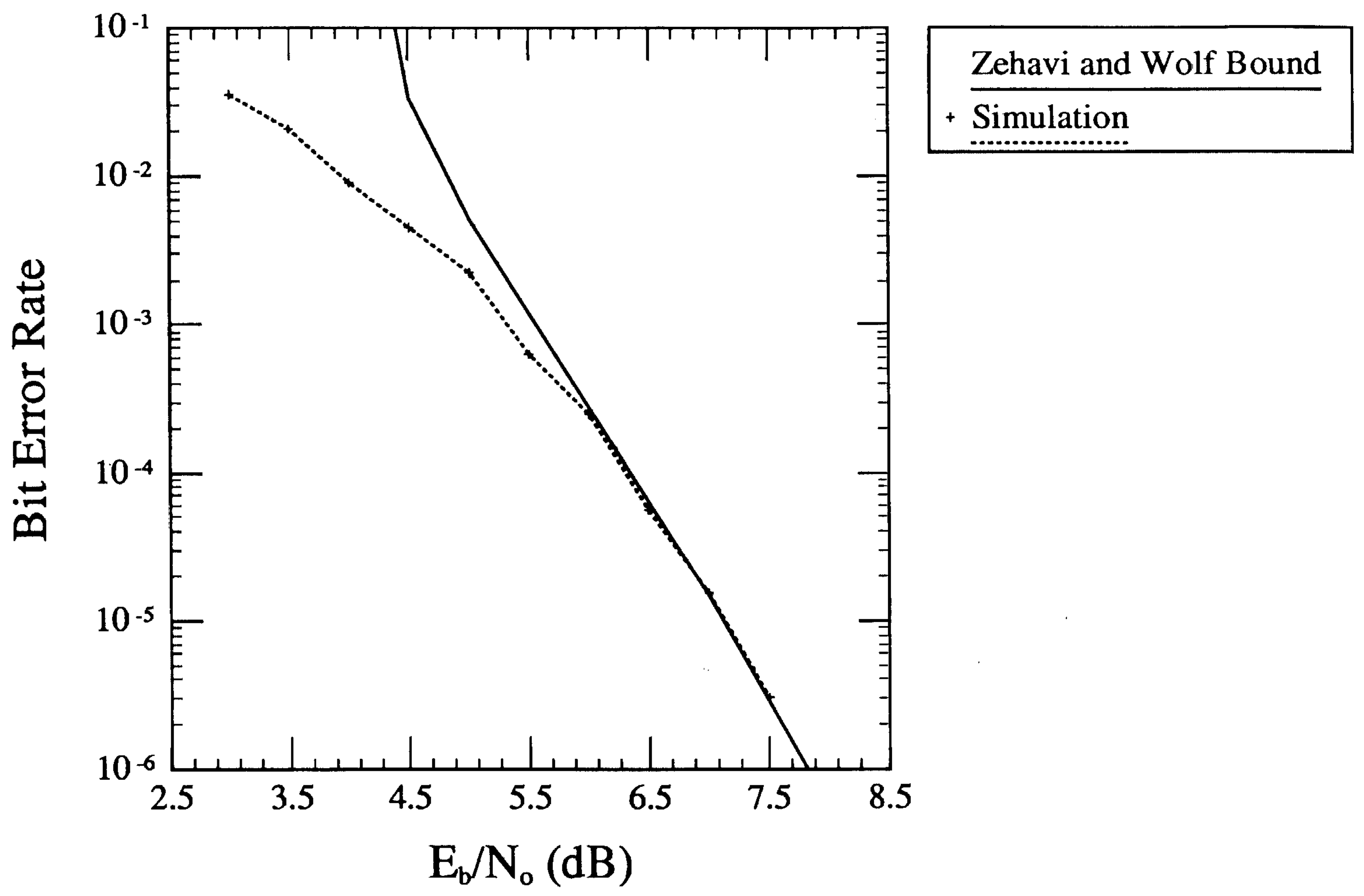


Figure 7: The Performance of the Ungerboeck 4 State, 8PSK, R=2.0 bits/T Trellis Code

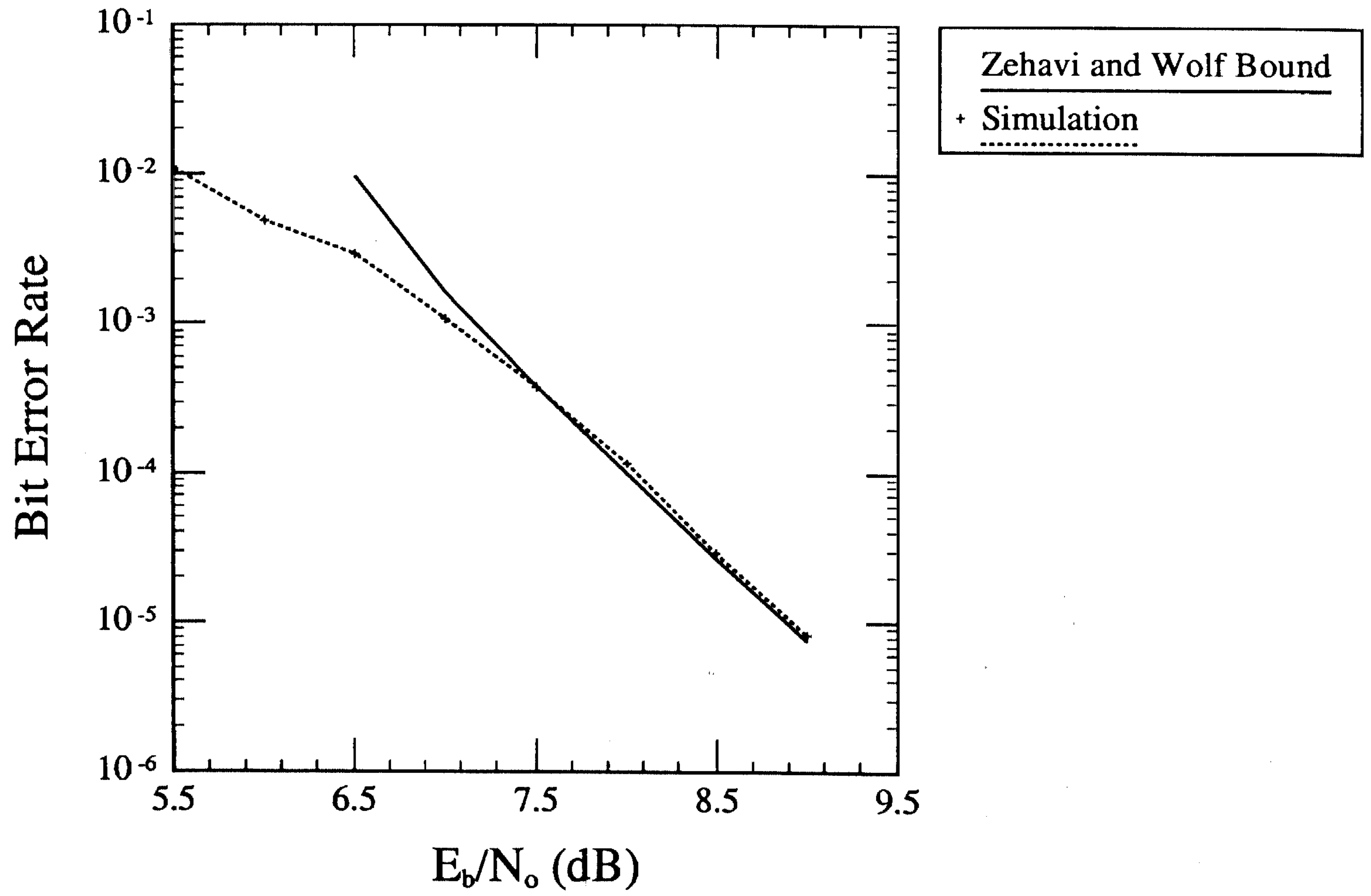


Figure 8: The Performance of a 4 State, 2x8PSK, R=2.5 bits/T Multi-D Trellis Code

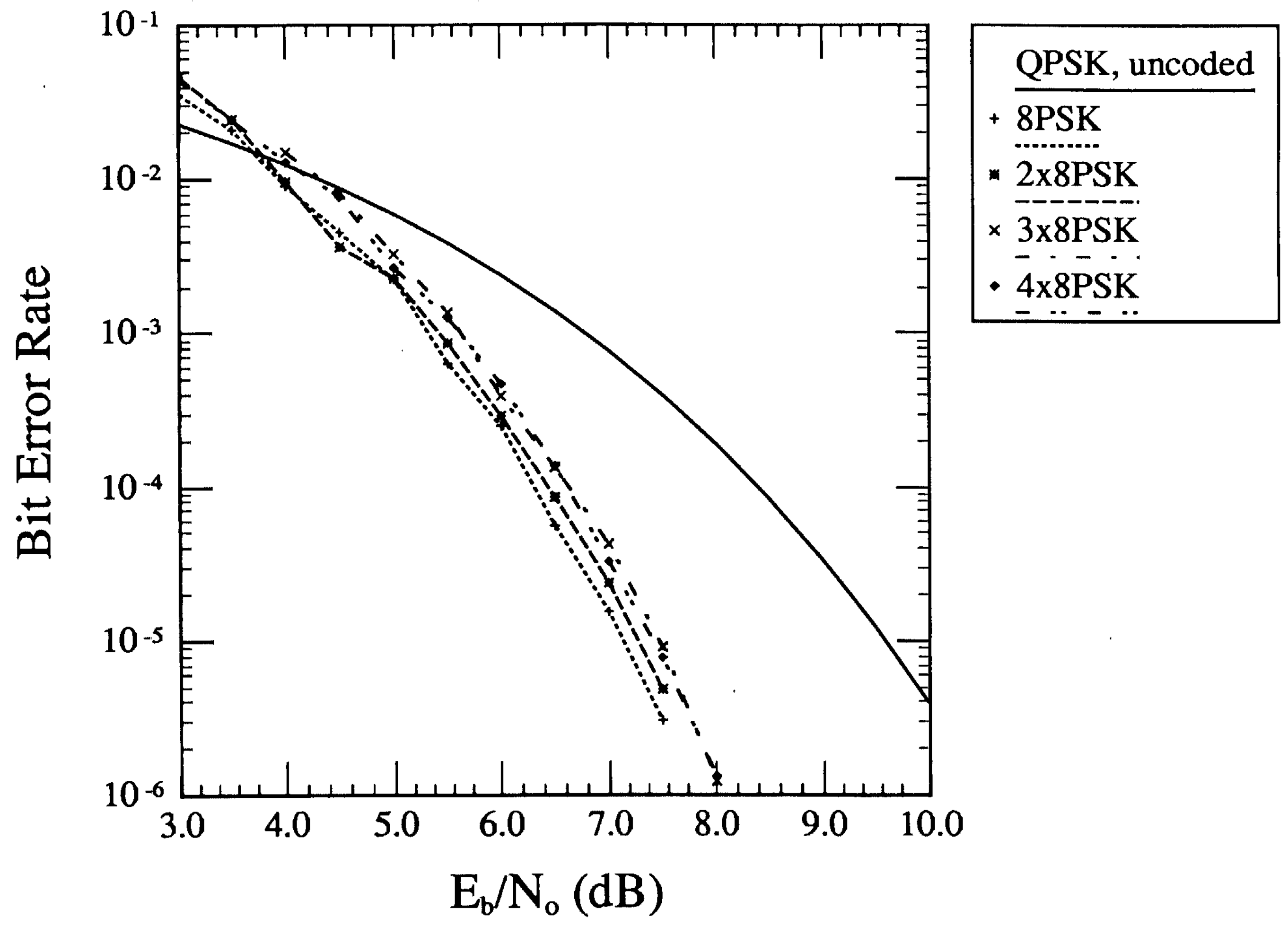


Figure 9: Simulation Results for Lx8PSK Codes with ν=2 and R=2.0 bits/T.

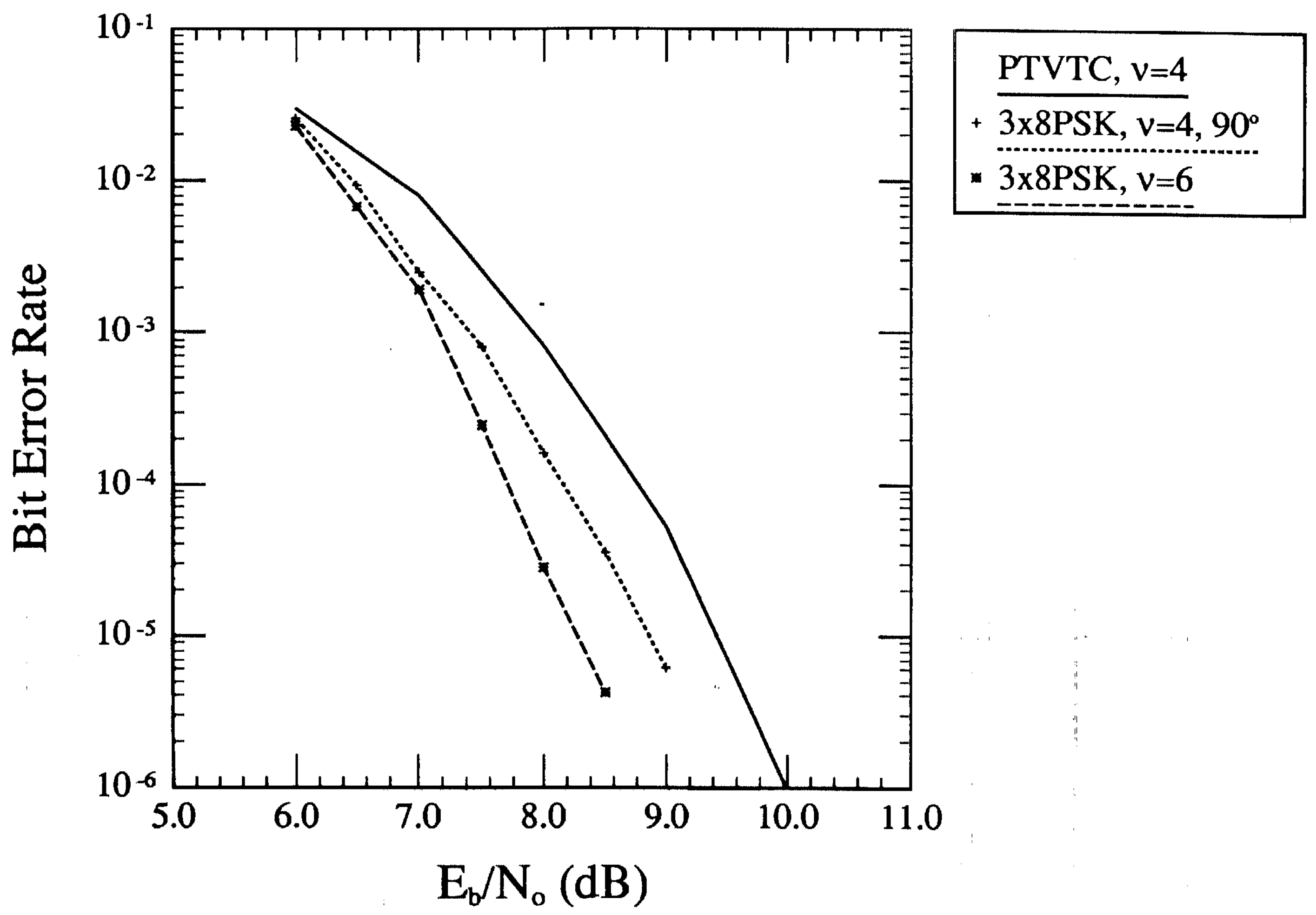


Figure 10: Simulation Results for Trellis Codes with R=2.67 bits/T.

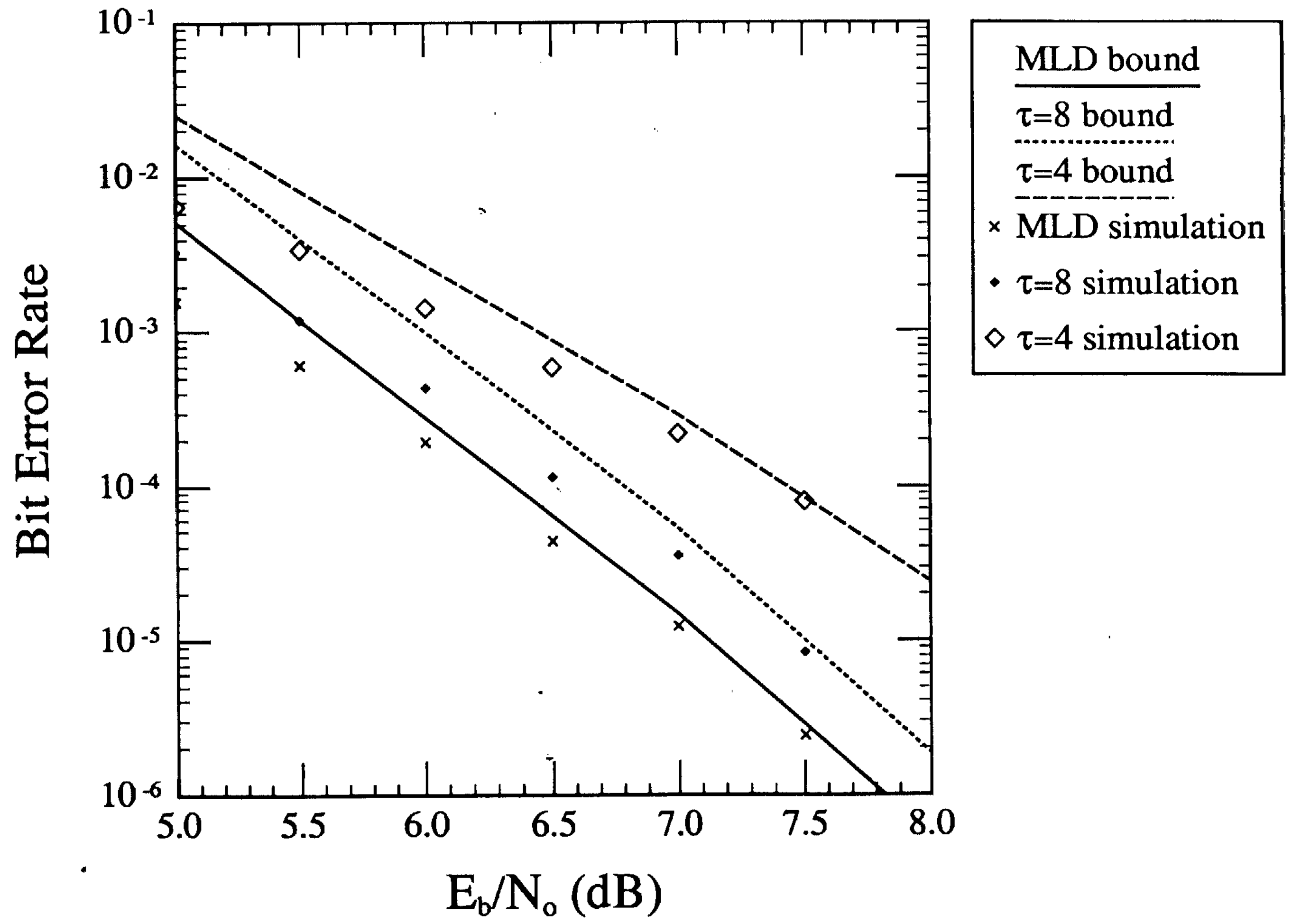


Figure 11: Truncation Length Results for the 4 State, 8PSK, Ungerboeck Code.

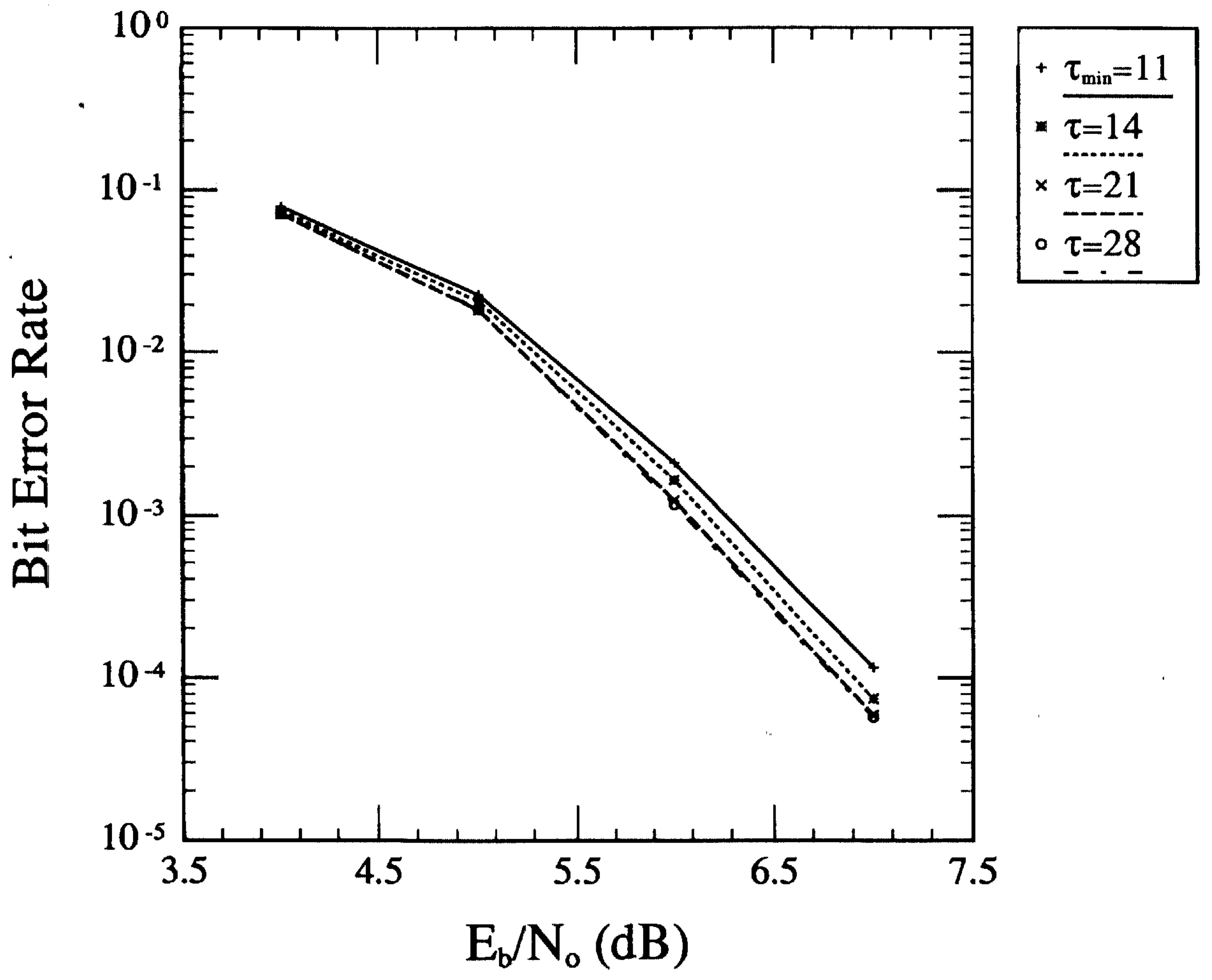


Figure 12: Truncation Length Results for the 16 State, 3x8PSK Multi-D Code with R=2.33bits/T..

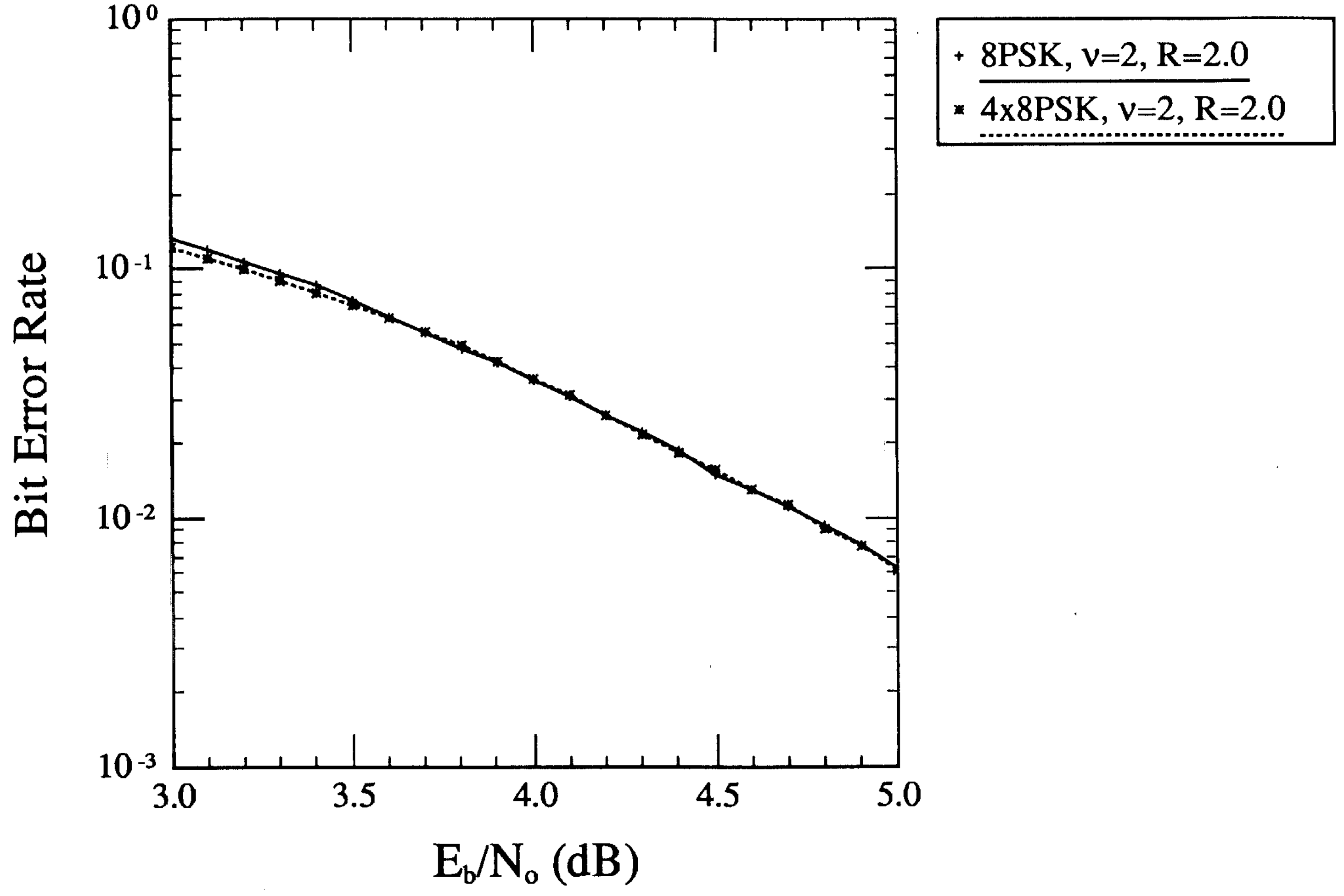


Figure 13: Symbol Error Rate Simulation Results for a RS(255,223) Outer Code.

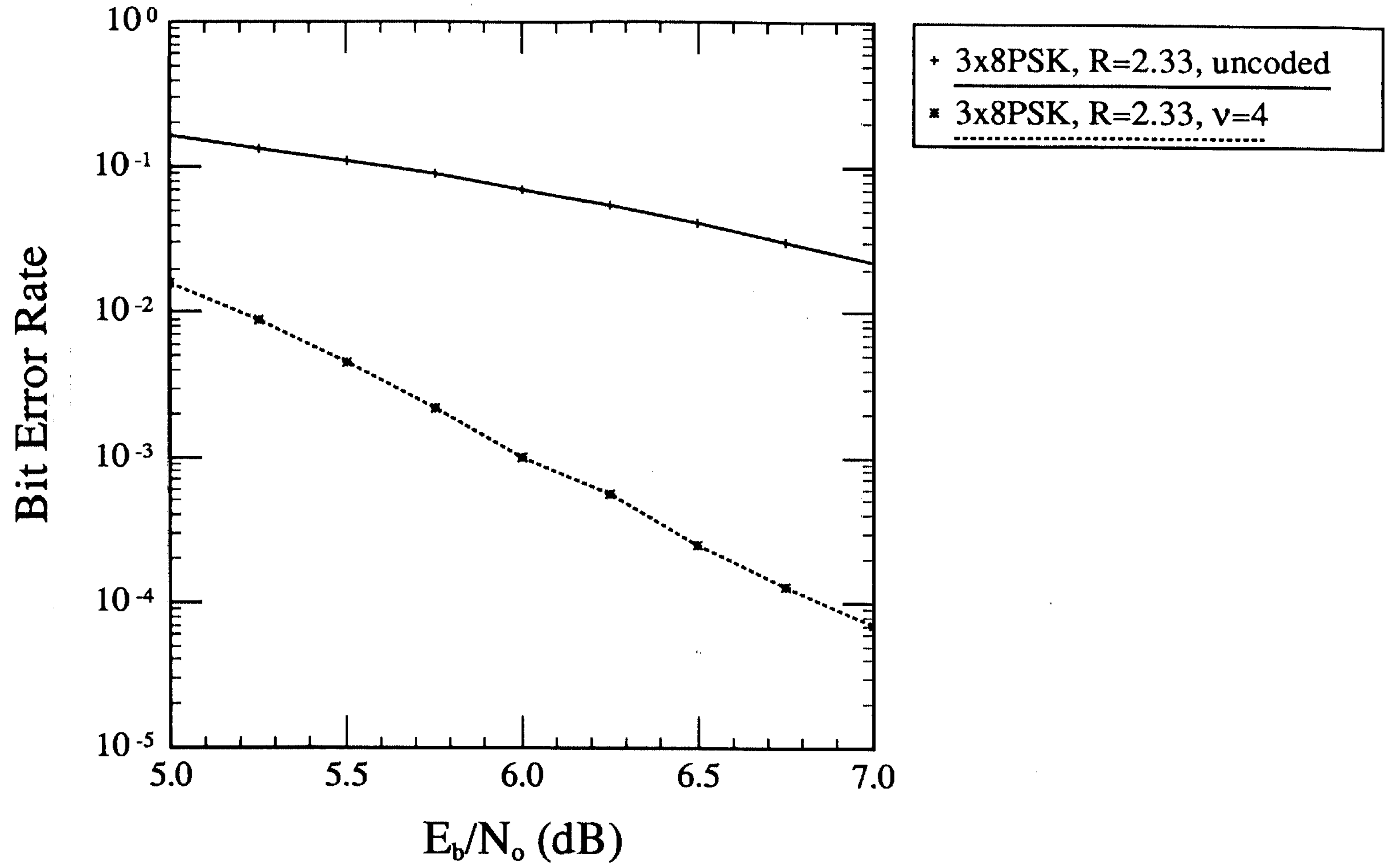


Figure 14: Symbol Error Rate Simulation Results for a RS(255,223) Outer Code.

7 Appendix I

This appendix contains the simulation results for a number of LxMPSK trellis codes. All table numbers refer to reference [3].

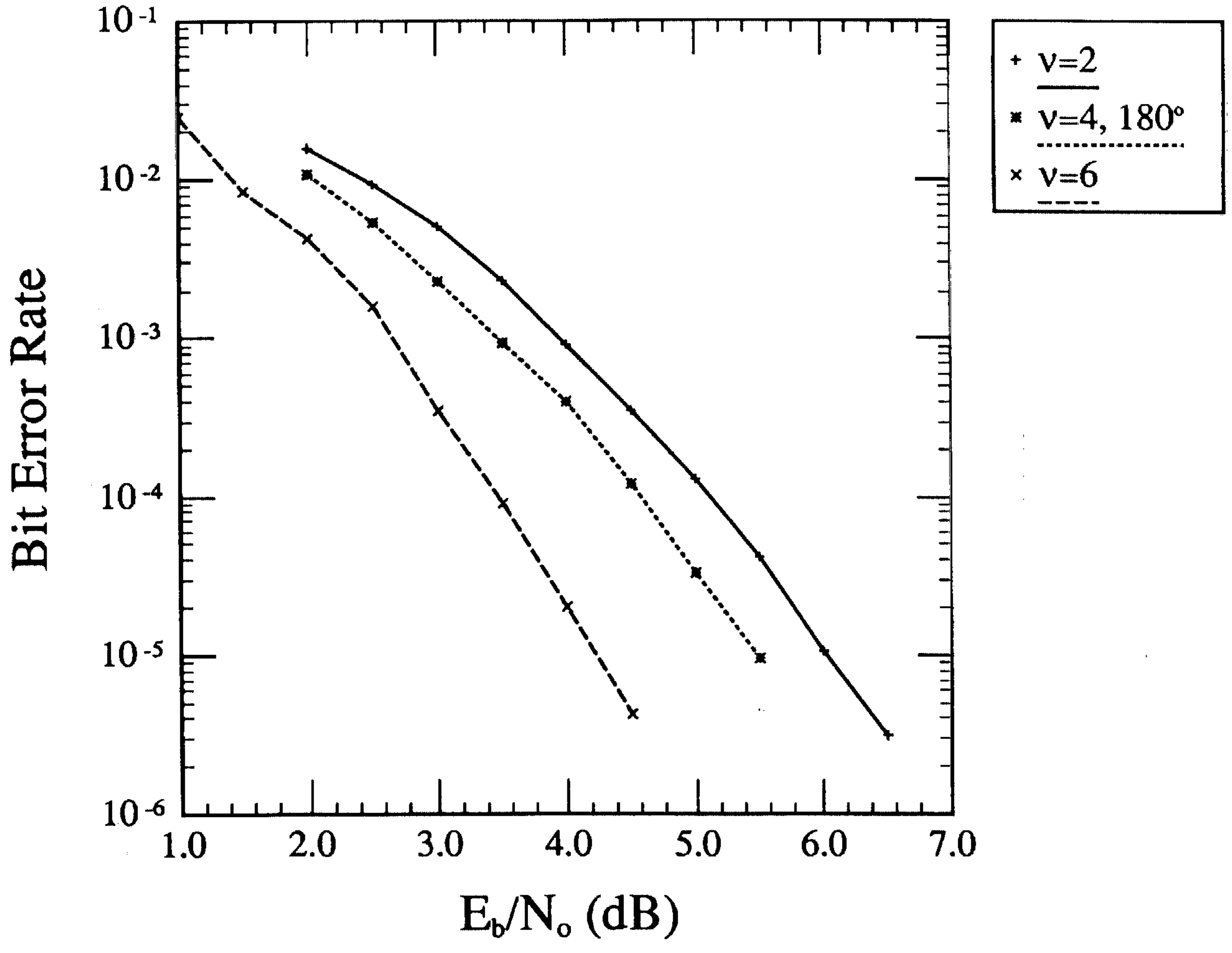


Figure A.1: 4PSK, R=1.0 bit/T Simulation Results (Table 11)

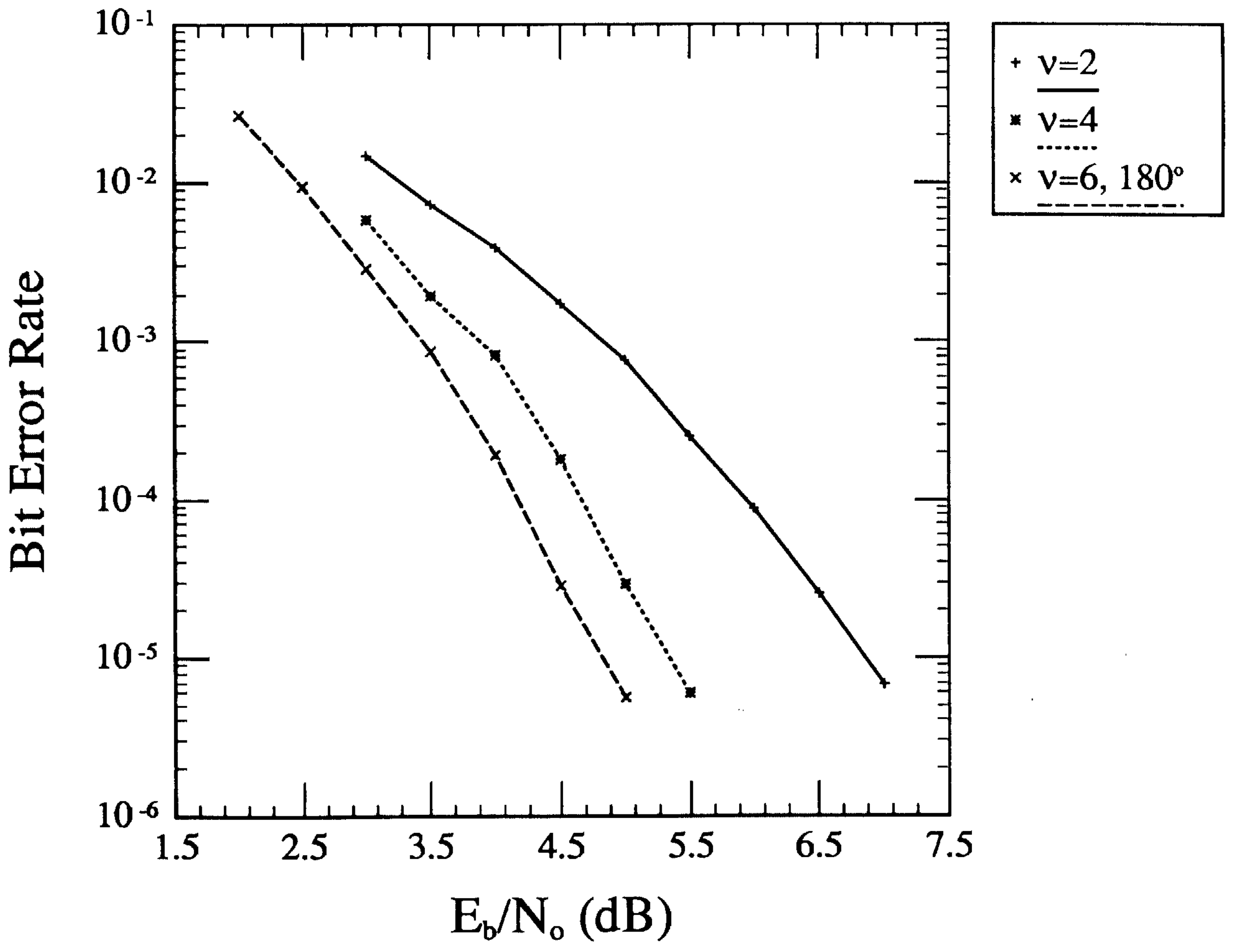


Figure A.2: 2x4PSK, R=1.5 bits/T Simulation Results (Table 12a)

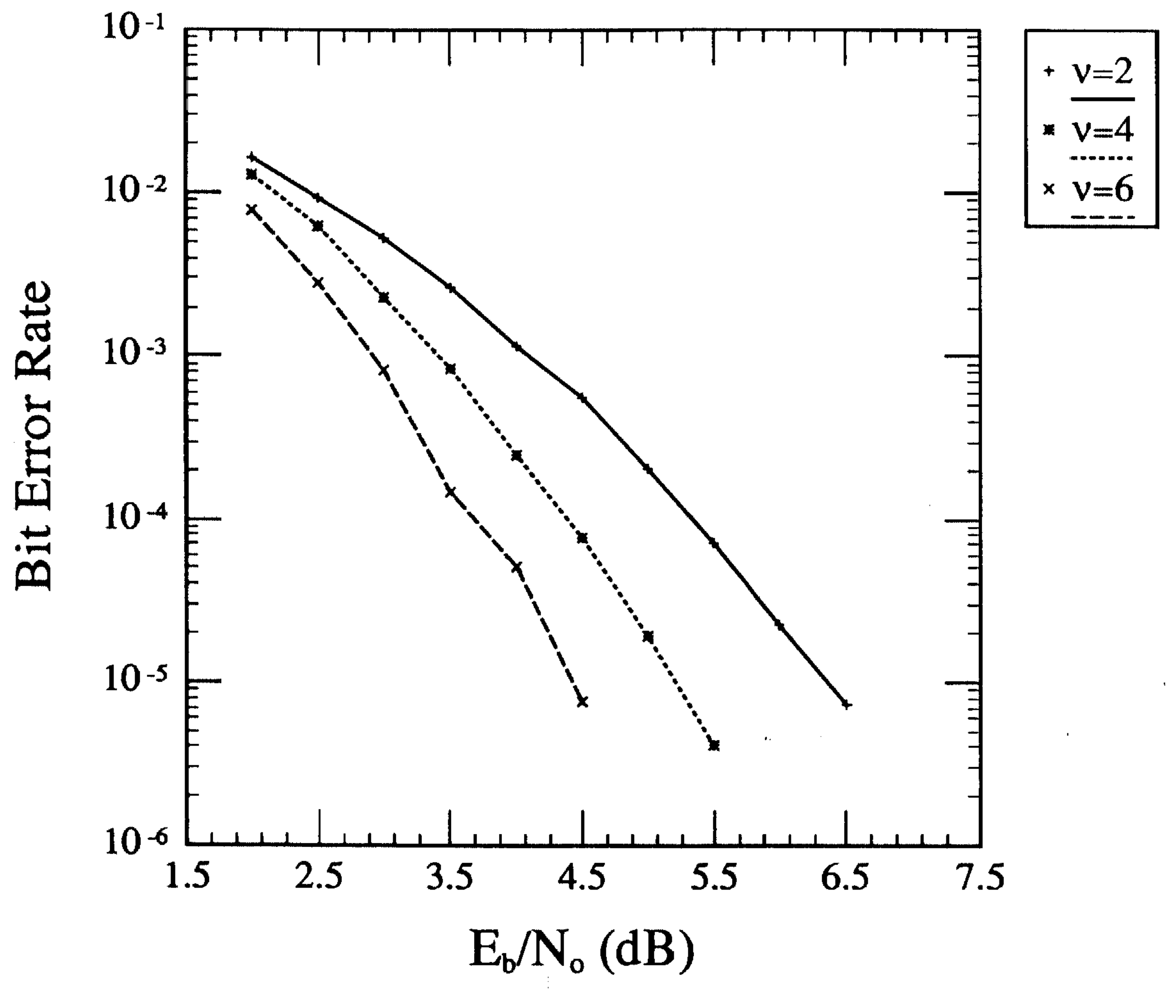


Figure A.3: 2x4PSK, R=1.0 bits/T Simulation Results (Table 12b)

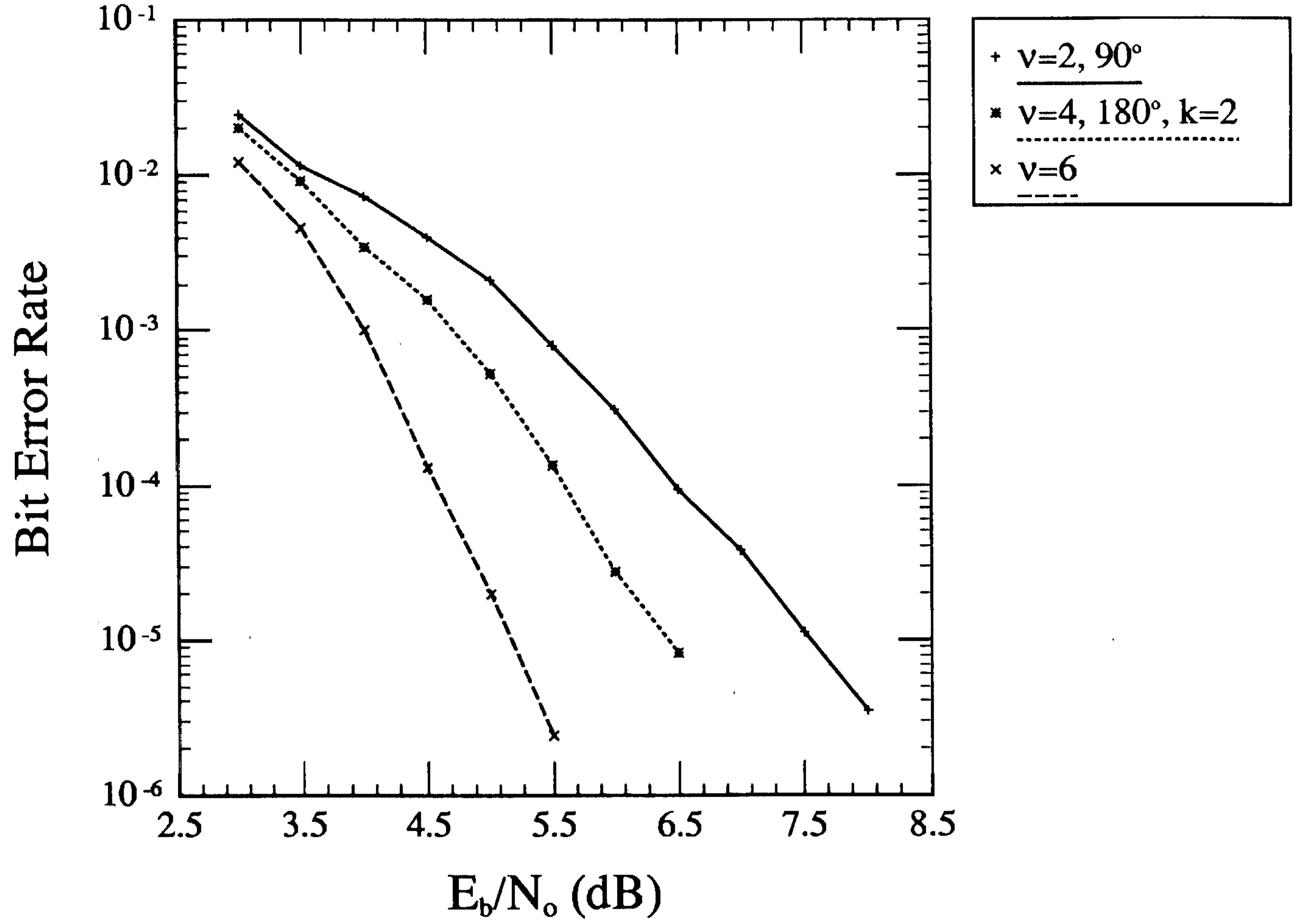


Figure A.4: 3x4PSK, R=1.67 bits/T Simulation Results (Table 13a)

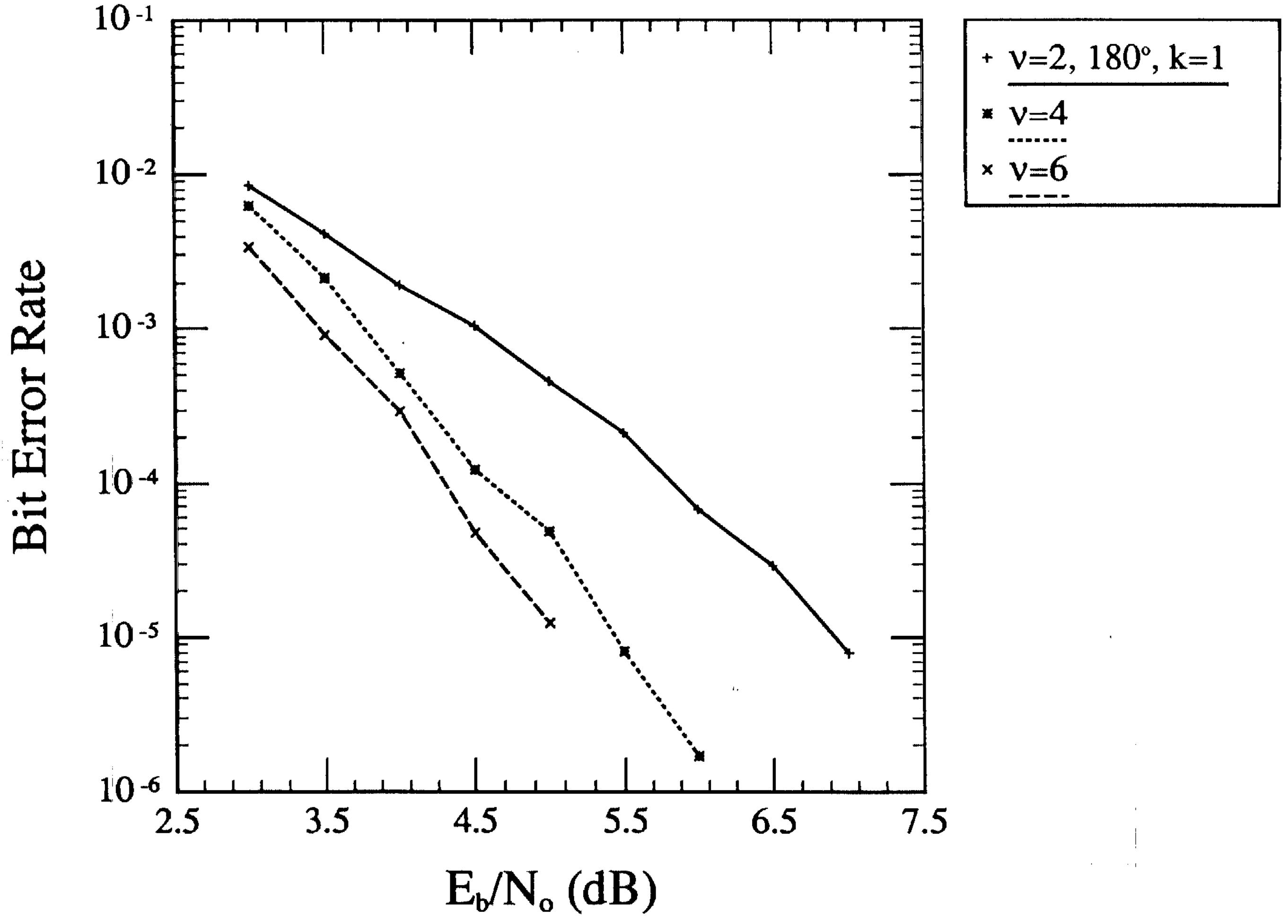


Figure A.5: 3x4PSK, R=1.33 bits/T Simulation Results (Table 13b)

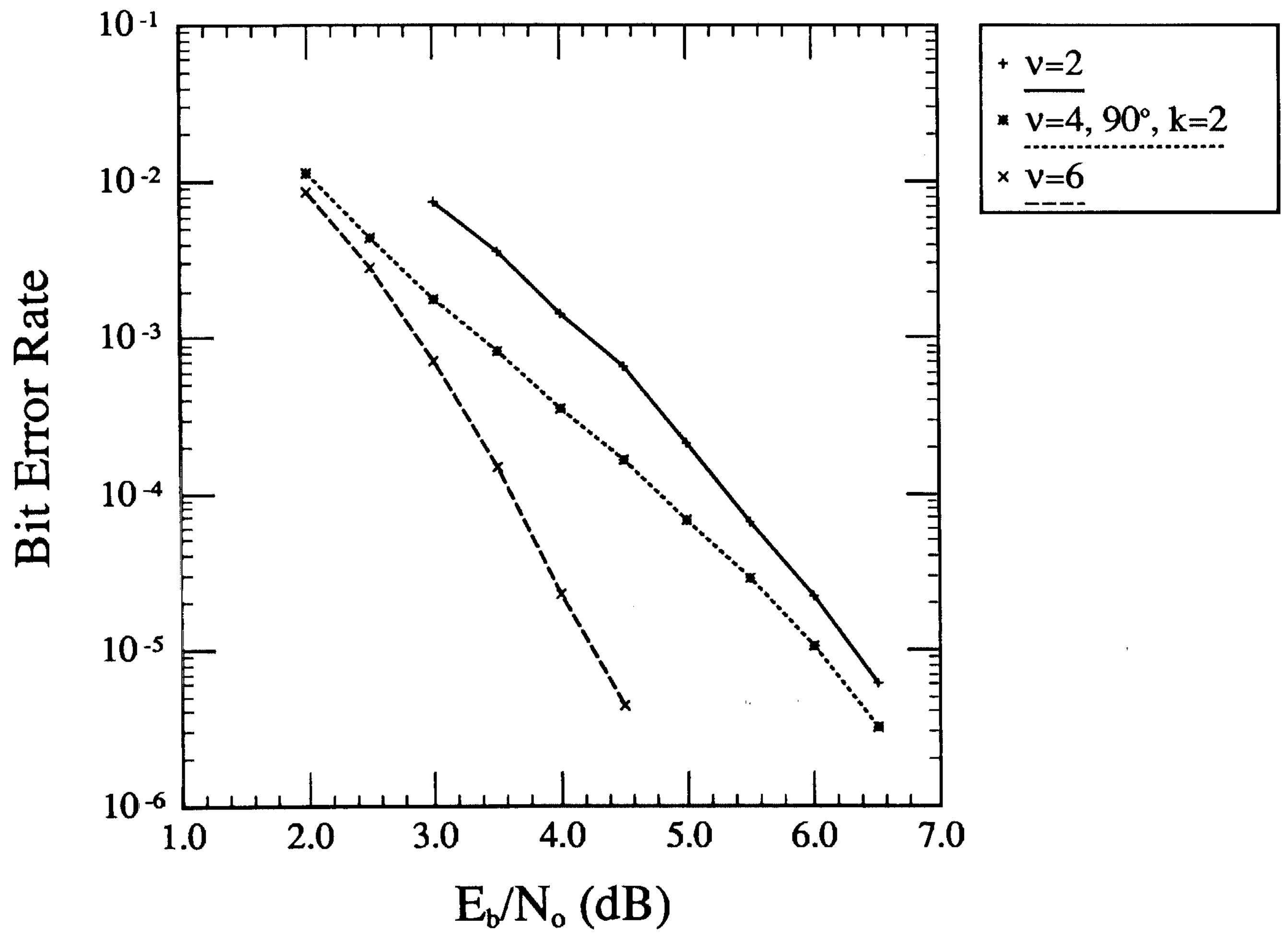


Figure A.6: 3x4PSK, R=1.00 bits/T Simulation Results (Table 13c)

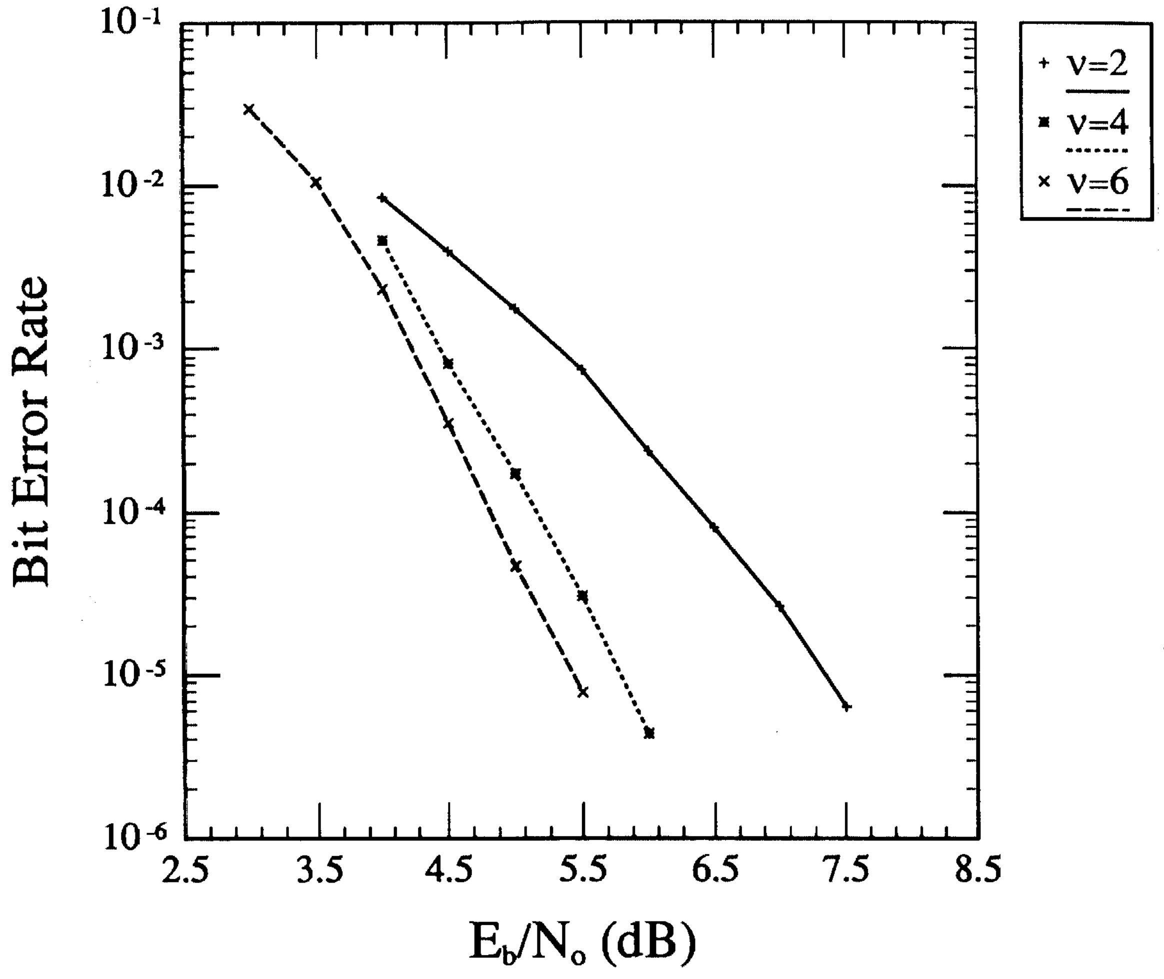


Figure A.7: 4x4PSK, R=1.75 bits/T Simulation Results (Table 14a)

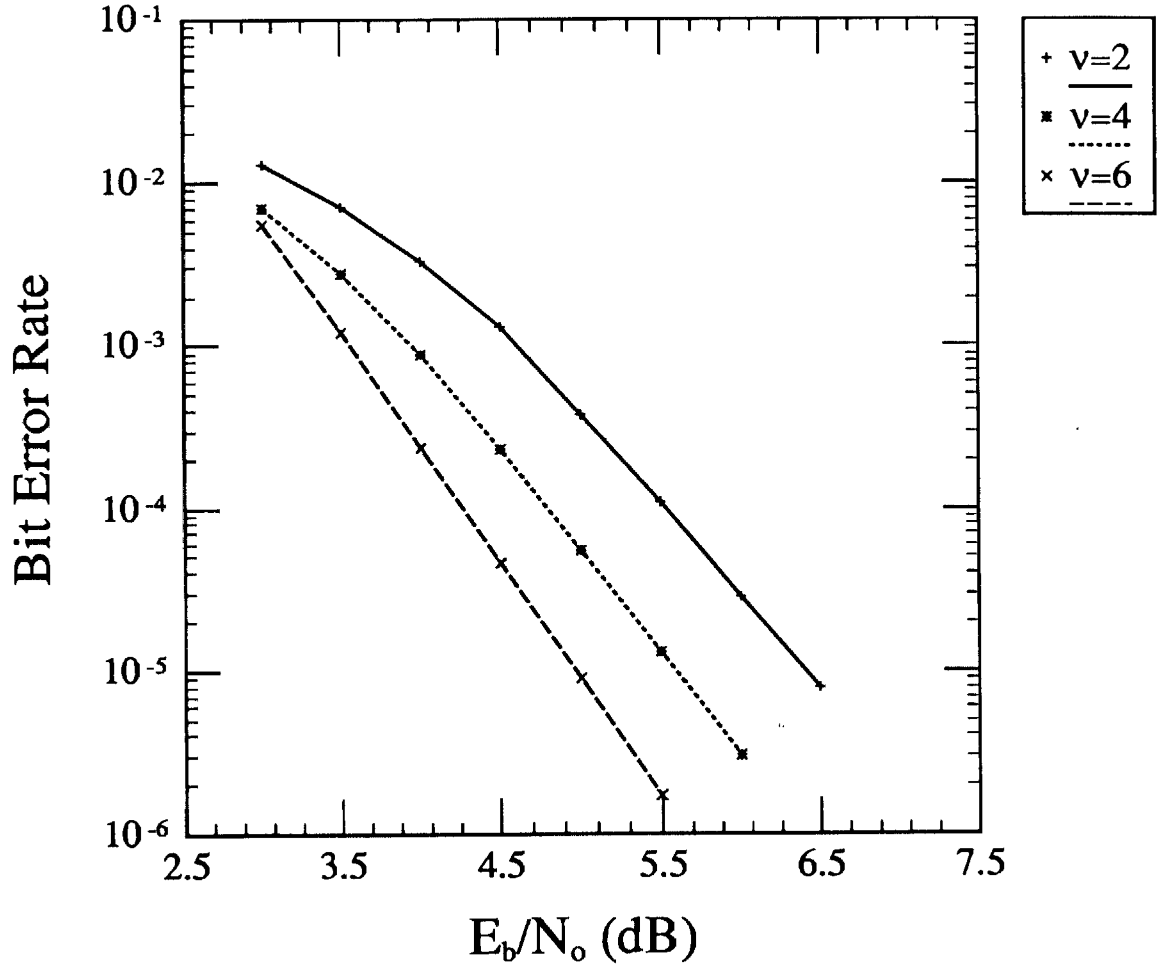


Figure A.8: 4x4PSK, R=1.50 bits/T Simulation Results (Table 14b)

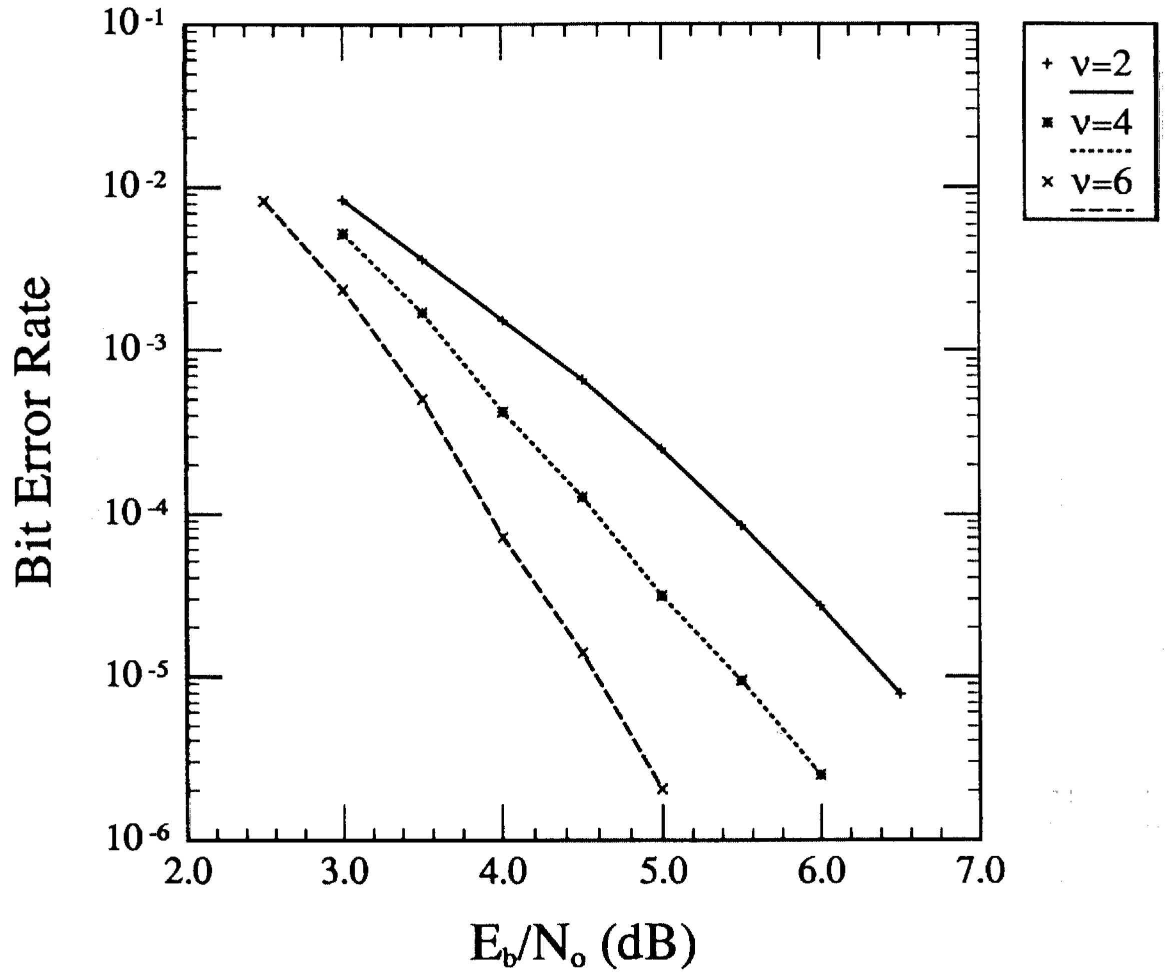


Figure A.9: 4x4PSK, R=1.25 bits/T Simulation Results (Table 14c)

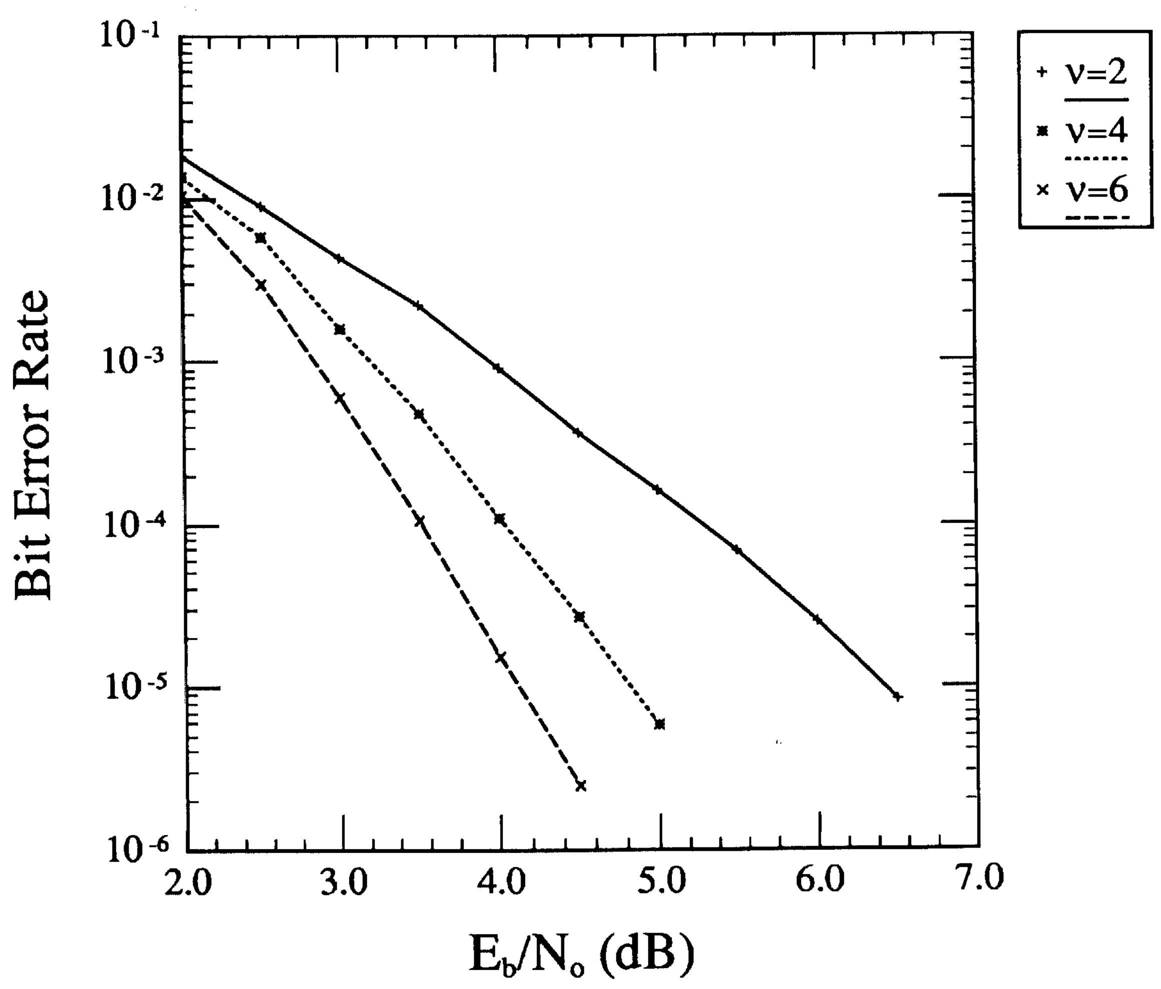


Figure A.10: 4x4PSK, R=1.00 bits/T Simulation Results (Table 14d)

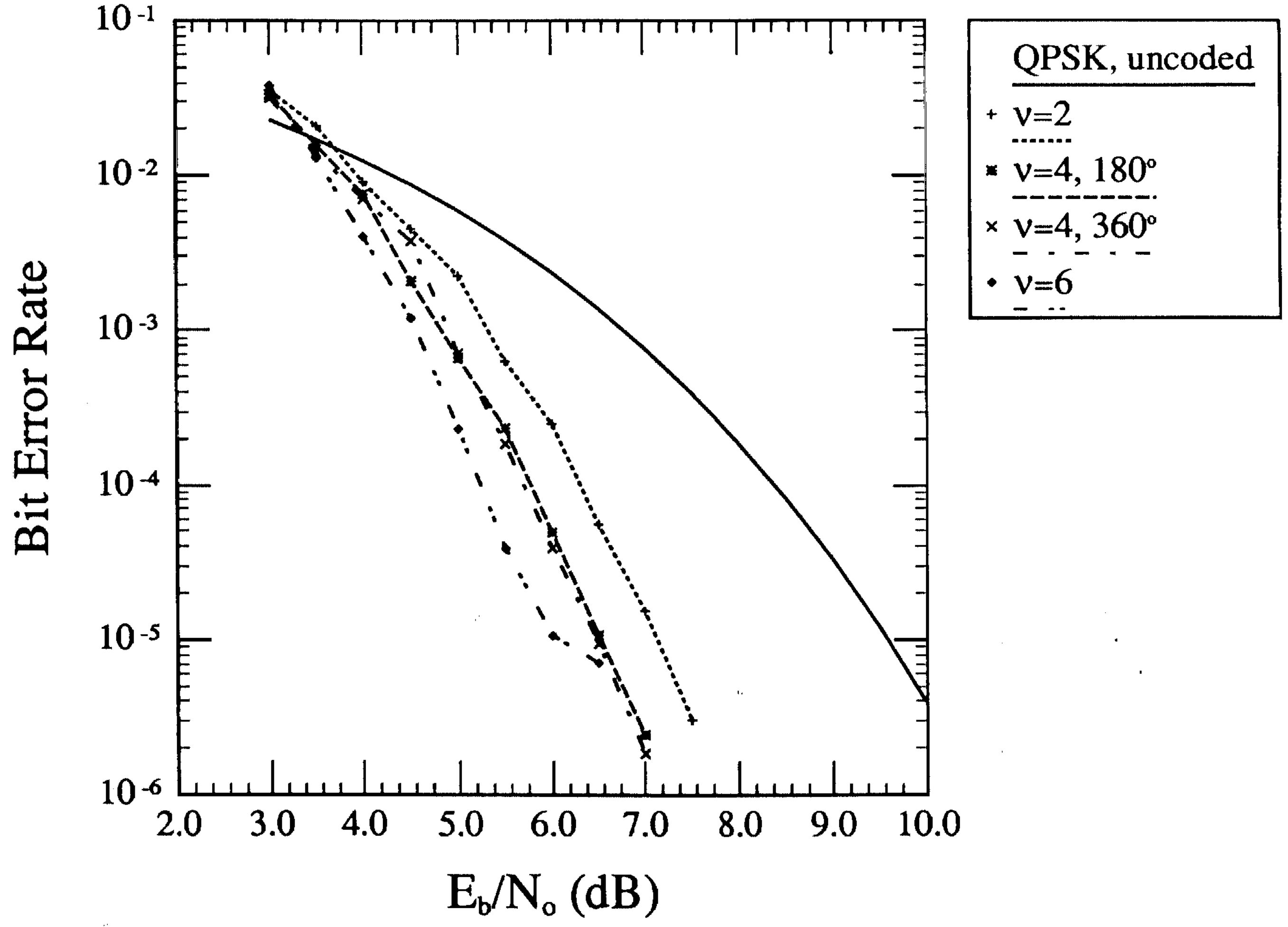


Figure A.11: 8PSK, R=2.00 bits/T Simulation Results (Table 15)

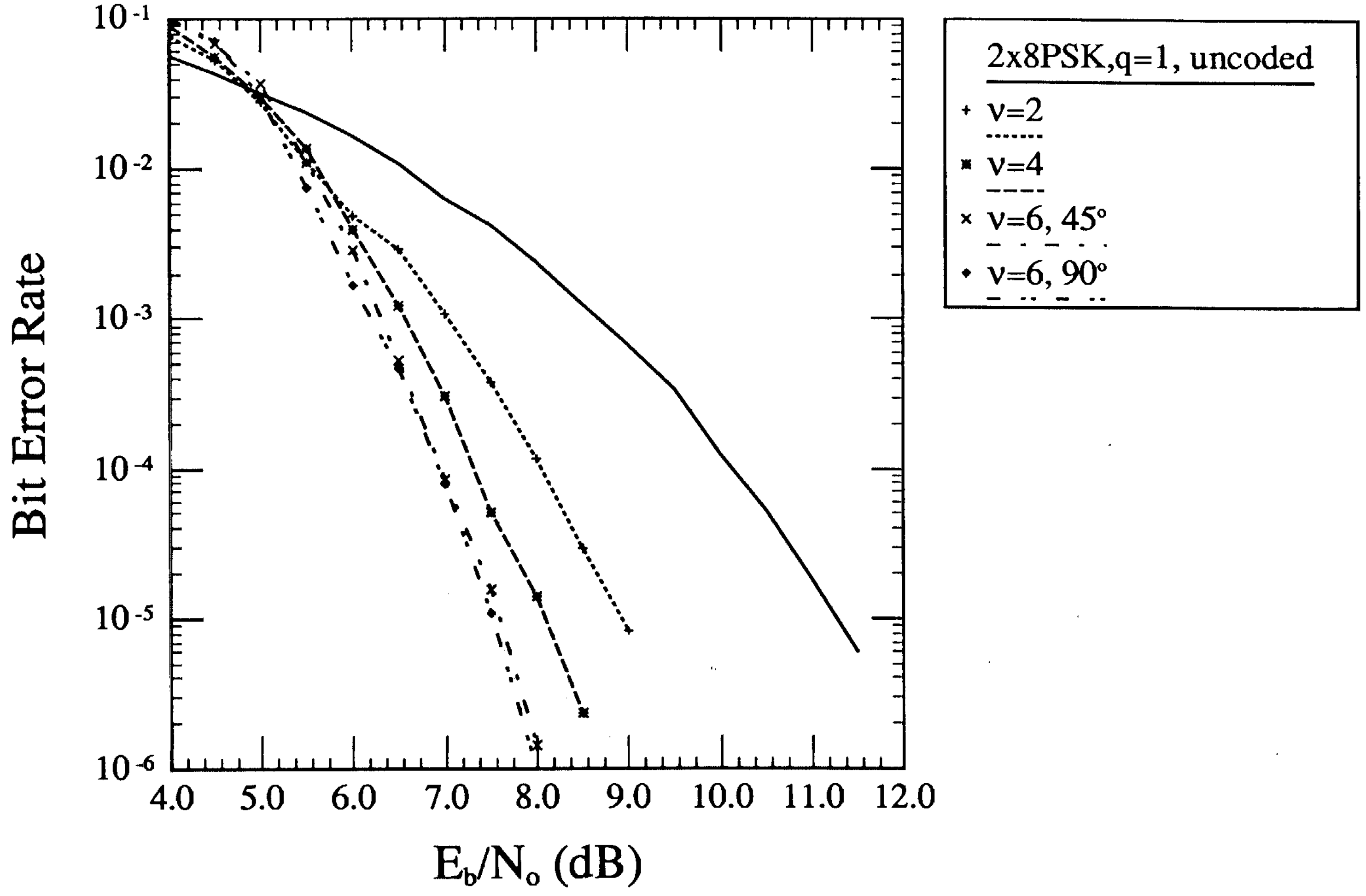


Figure A.12: 2x8PSK, R=2.50 bits/T Simulation Results (Table 16a)

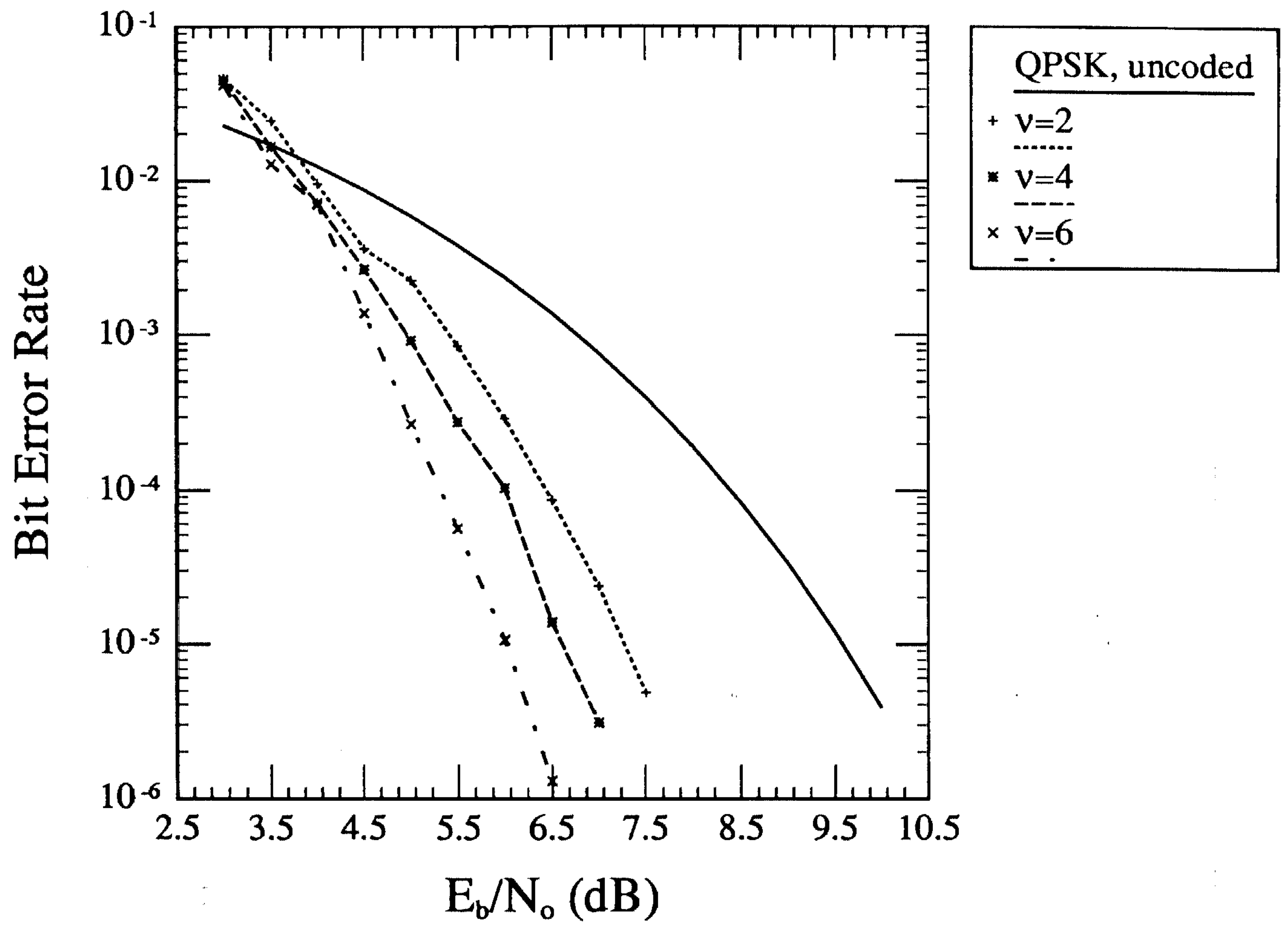


Figure A.13: 2x8PSK, R=2.00 bits/T Simulation Results (Table 16b)

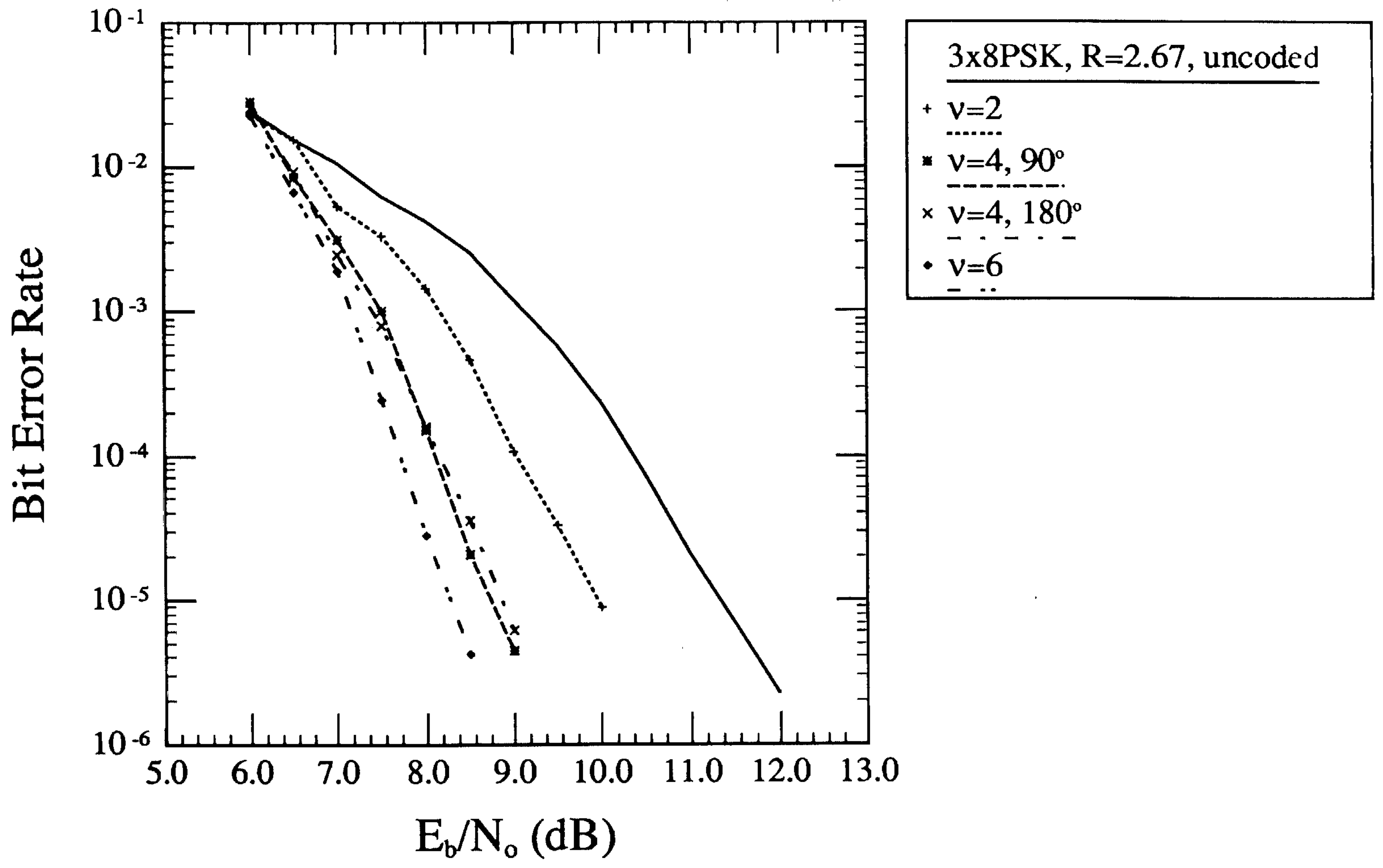


Figure A.14: 3x8PSK, R=2.67 bits/T Simulation Results (Table 17a)

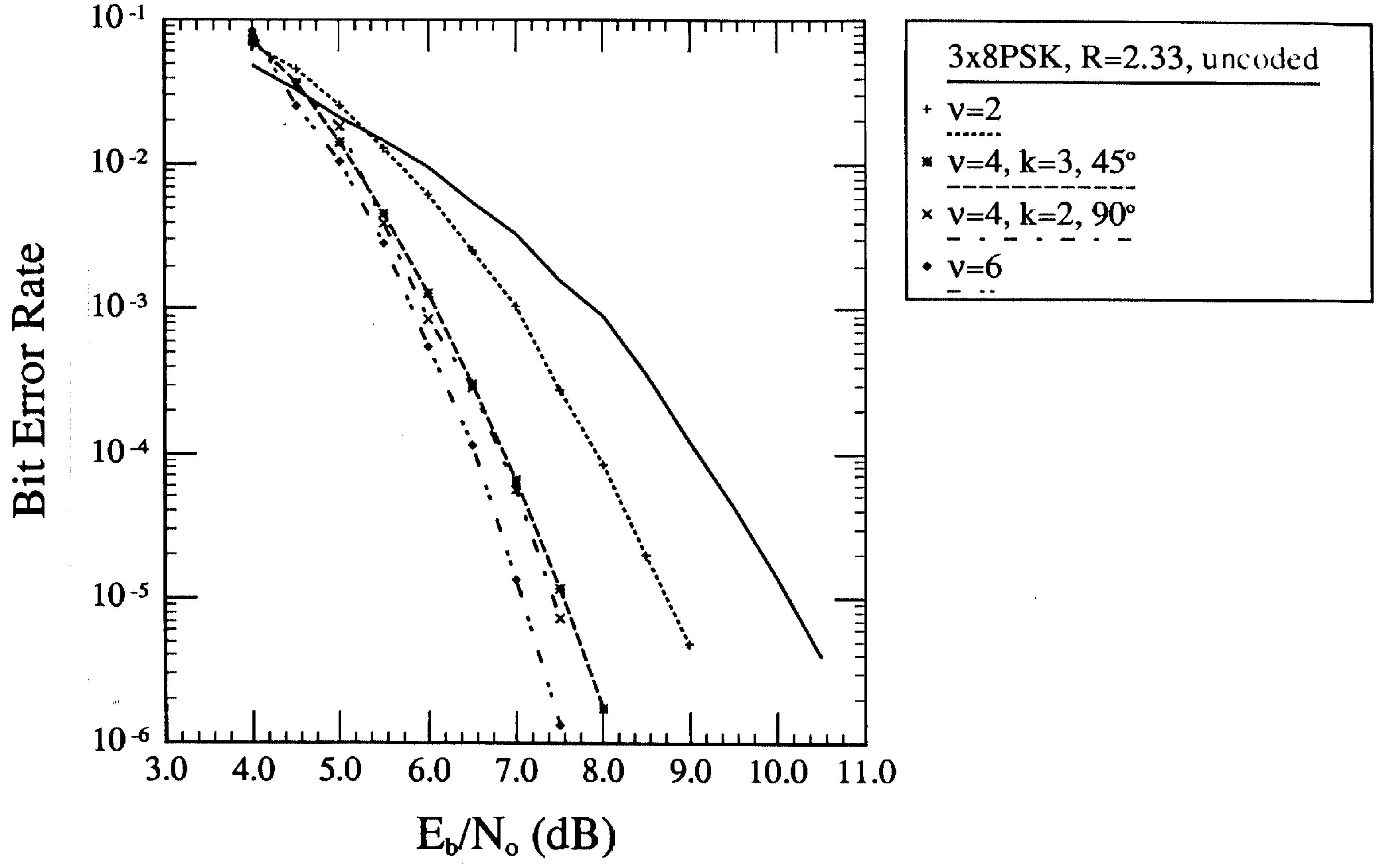


Figure A.15: 3x8PSK, R=2.33 bits/T Simulation Results (Table 17b)

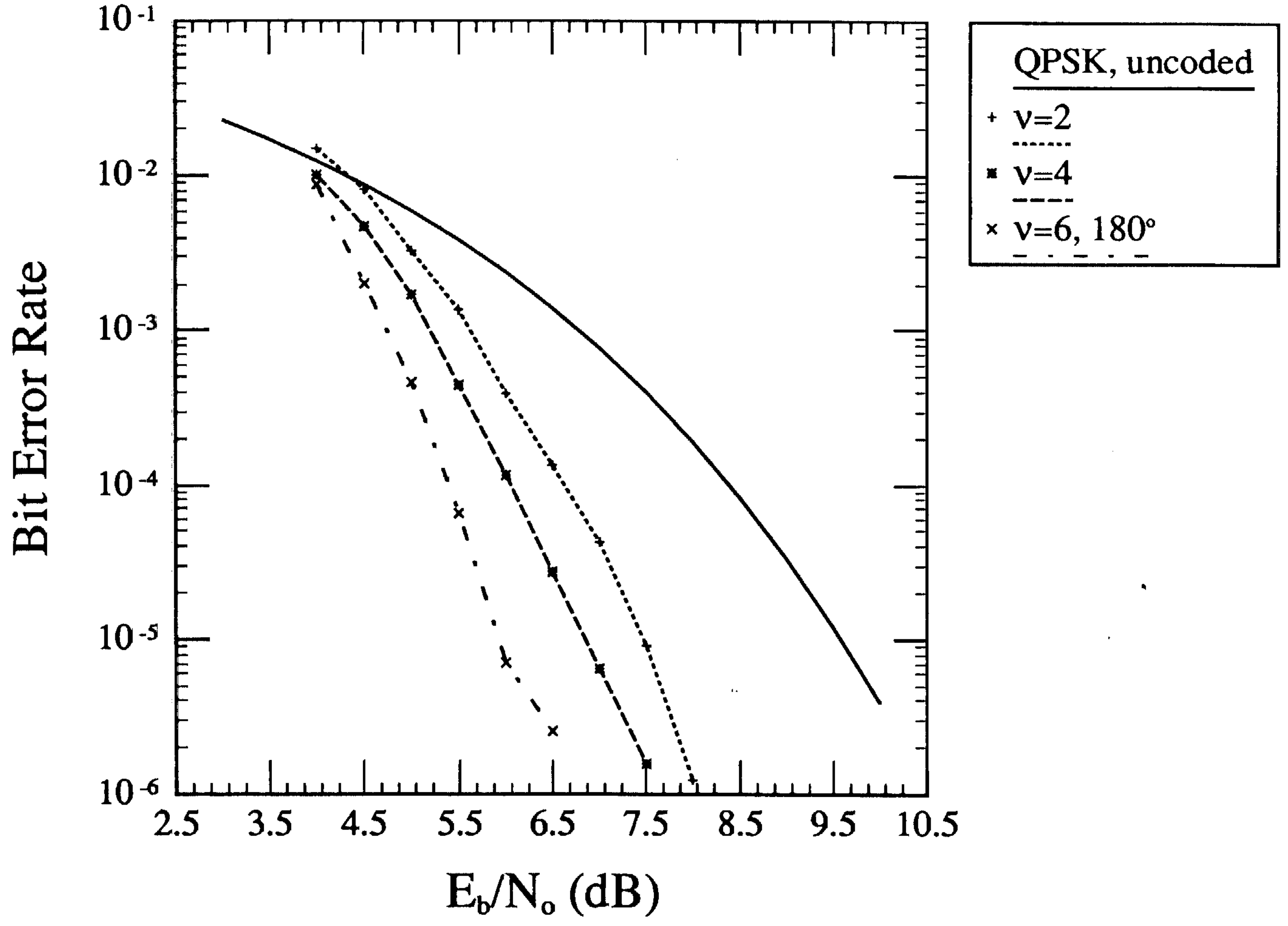


Figure A.16: 3x8PSK, R=2.00 bits/T Simulation Results (Table 17c)

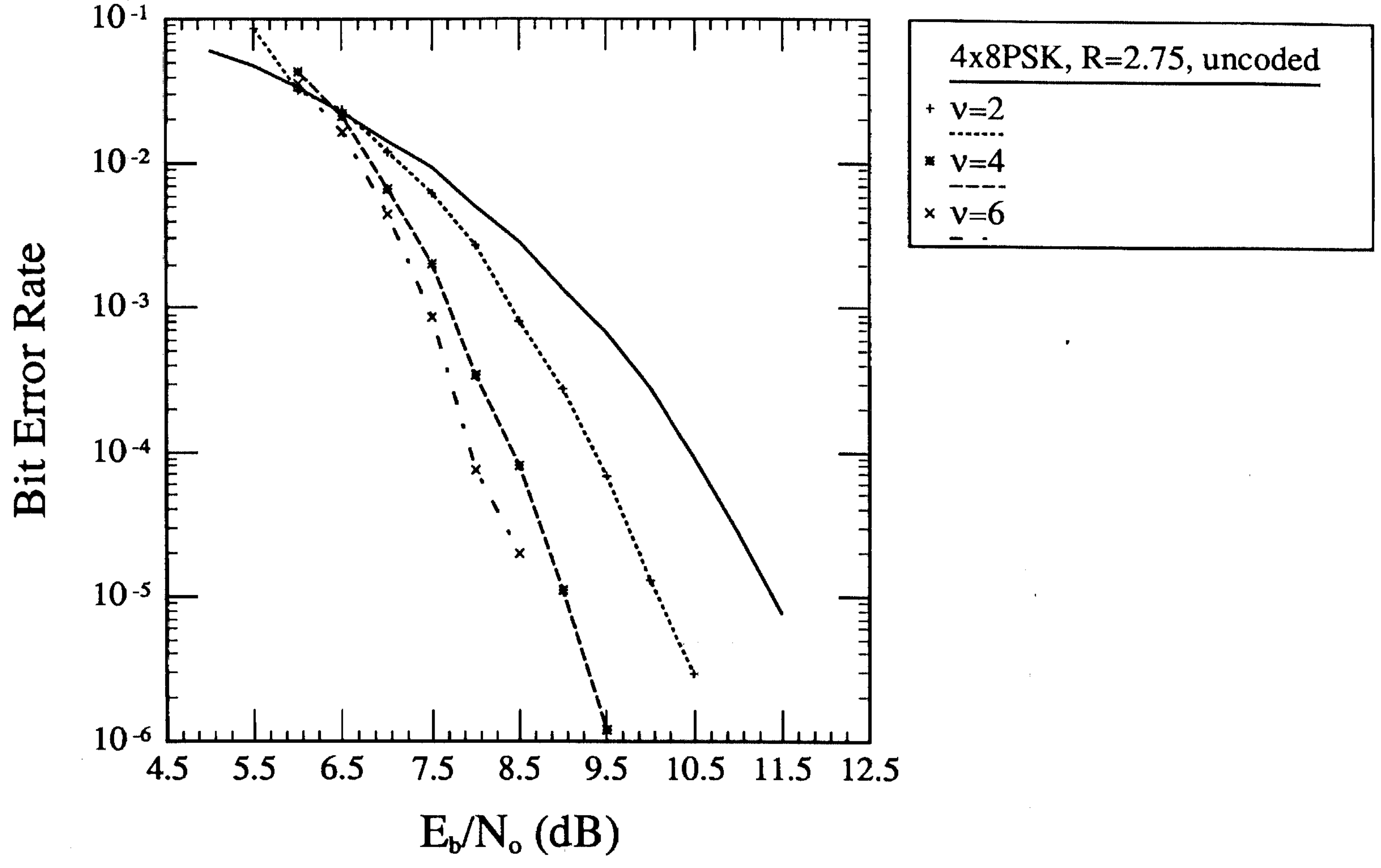


Figure A.17: 4x8PSK, R=2.75 bits/T Simulation Results (Table 18a)

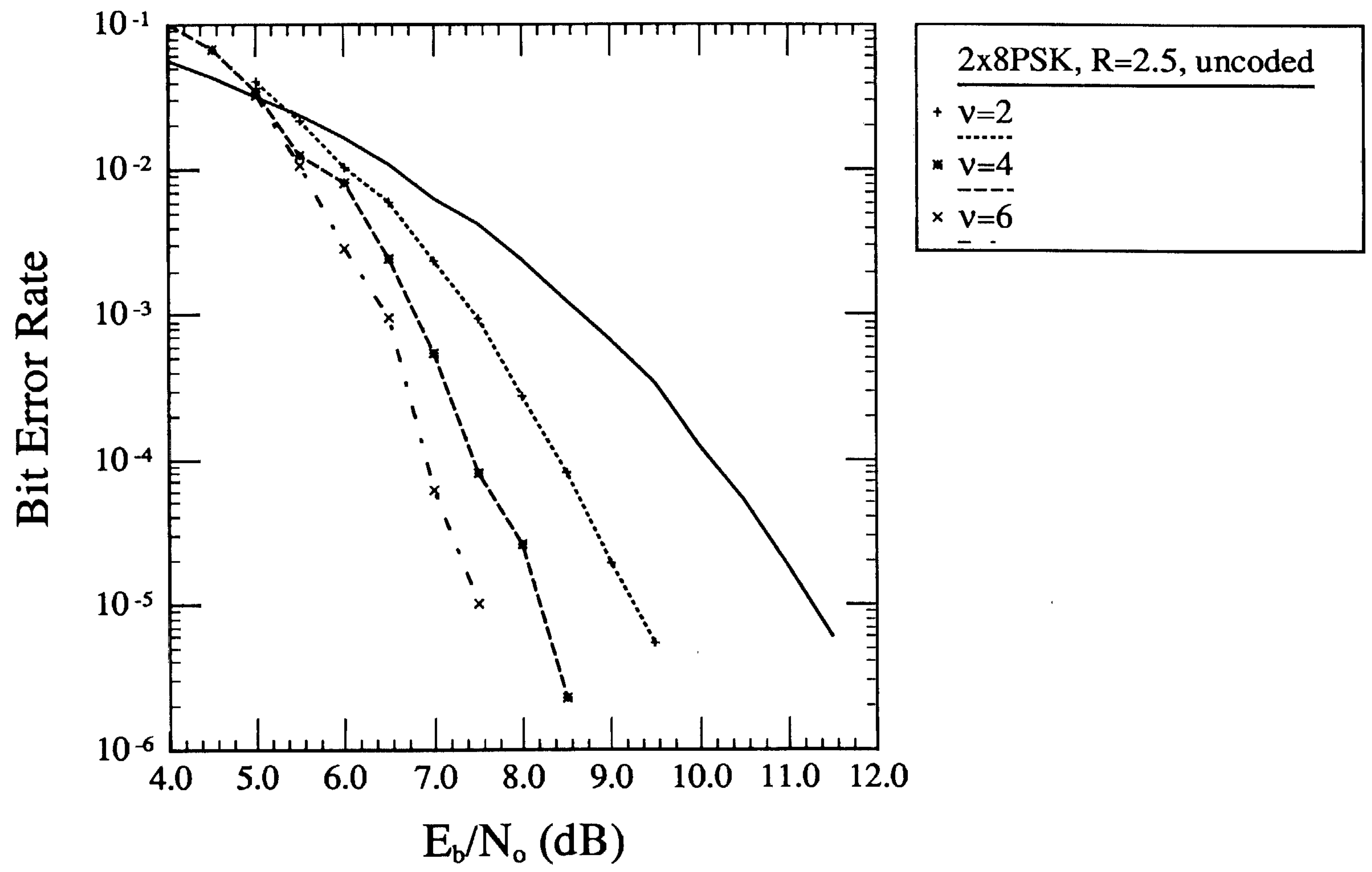


Figure A.18: 4x8PSK, R=2.50 bits/T Simulation Results (Table 18b)

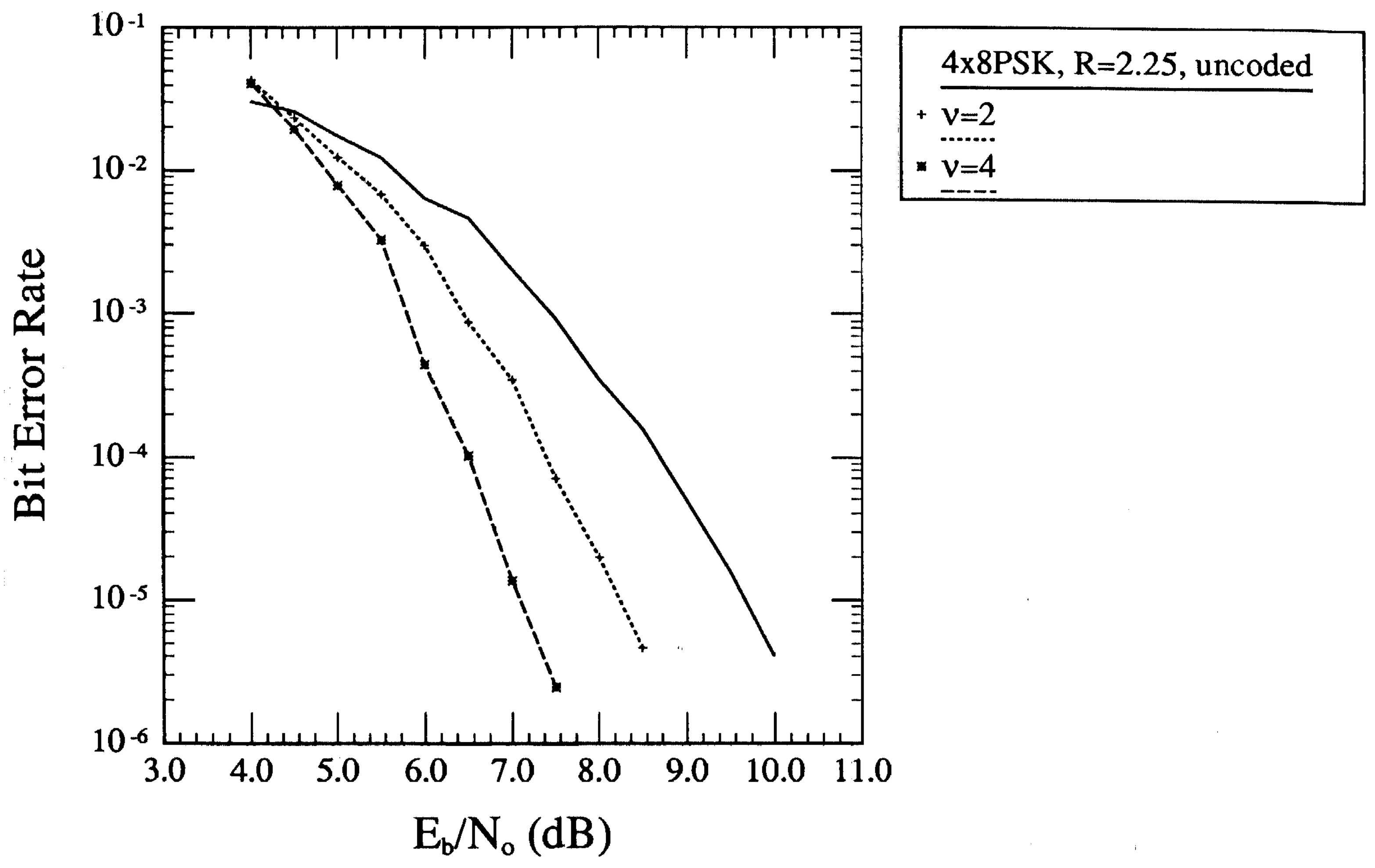


Figure A.19: 4x8PSK, R=2.25 bits/T Simulation Results (Table 18c)

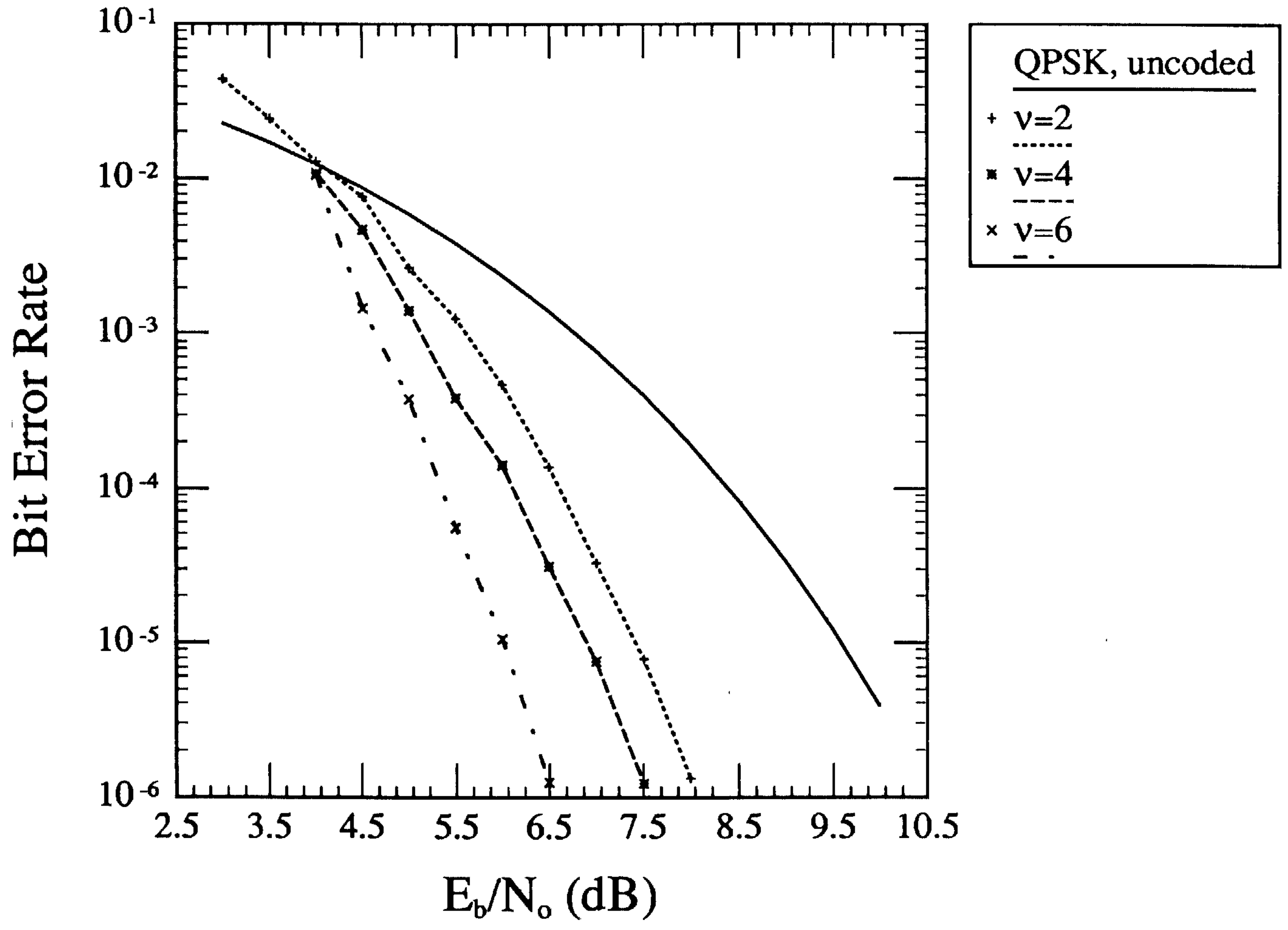


Figure A.20: 4x8PSK, R=2.00 bits/T Simulation Results (Table 18d)

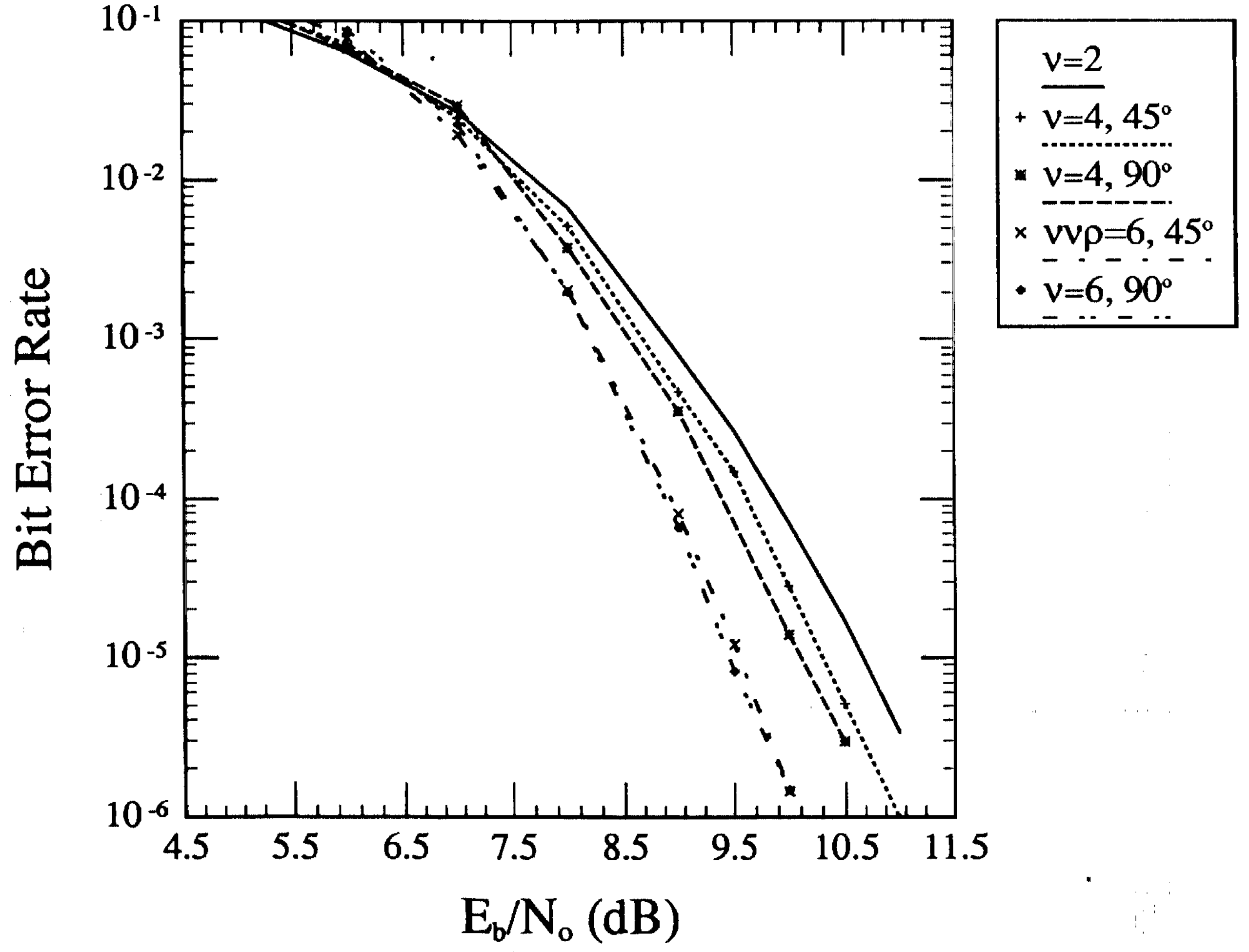


Figure A.21: 16PSK, R=3.00 bits/T Simulation Results (Table 19)

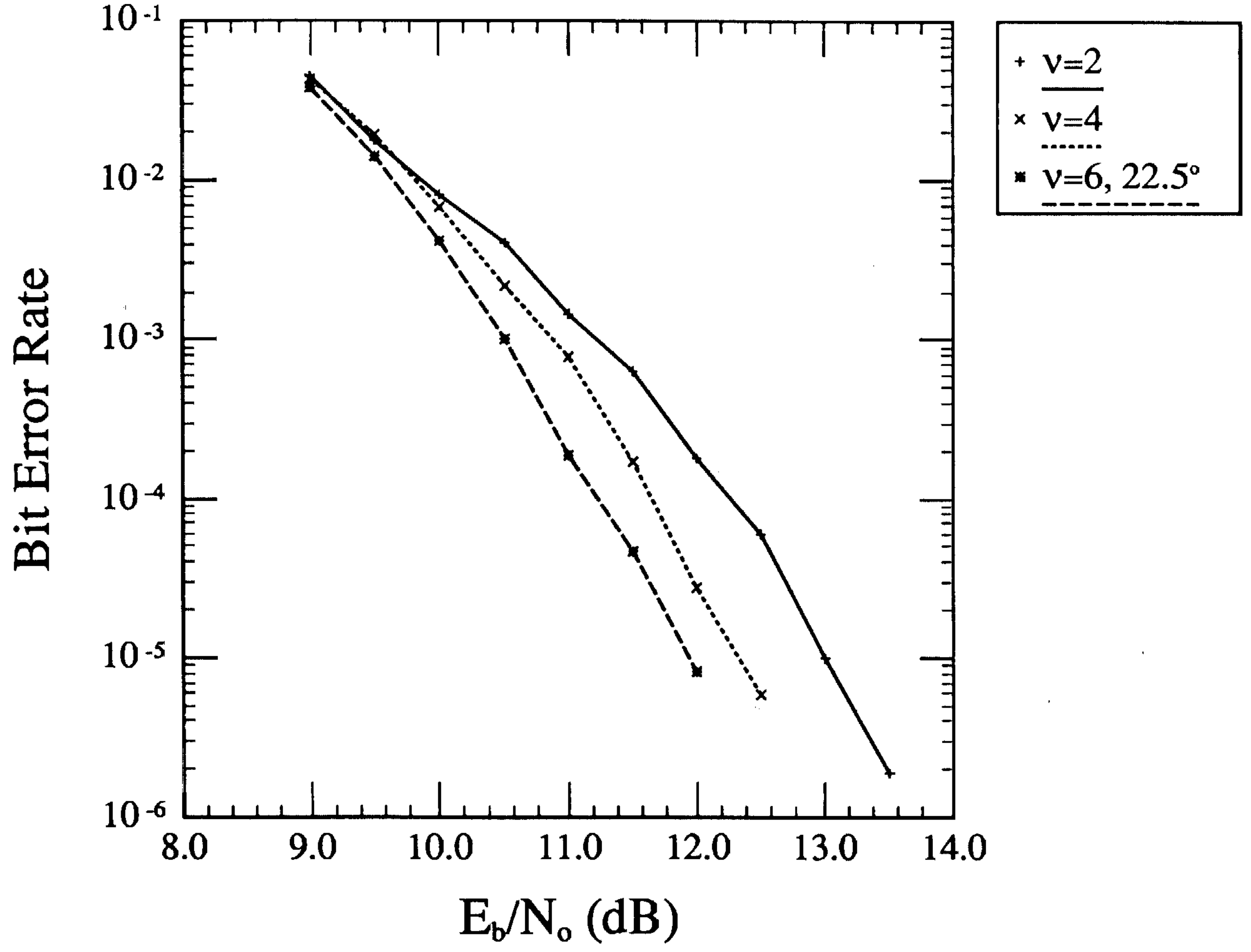


Figure A.22: 2x16PSK, R=3.50 bits/T Simulation Results (Table 20a)

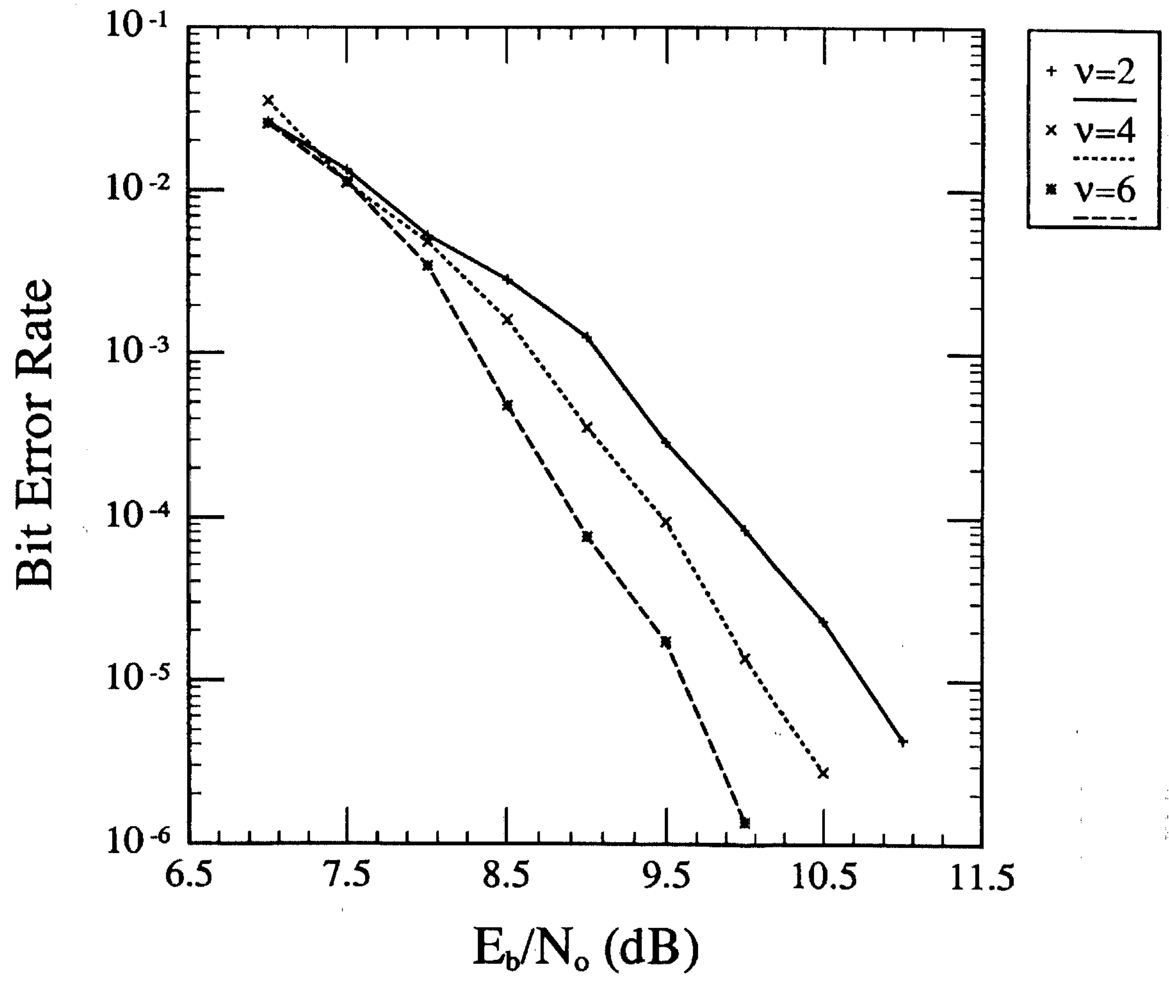


Figure A.23: 2x16PSK, R=3.00 bits/T Simulation Results (Table 20b)

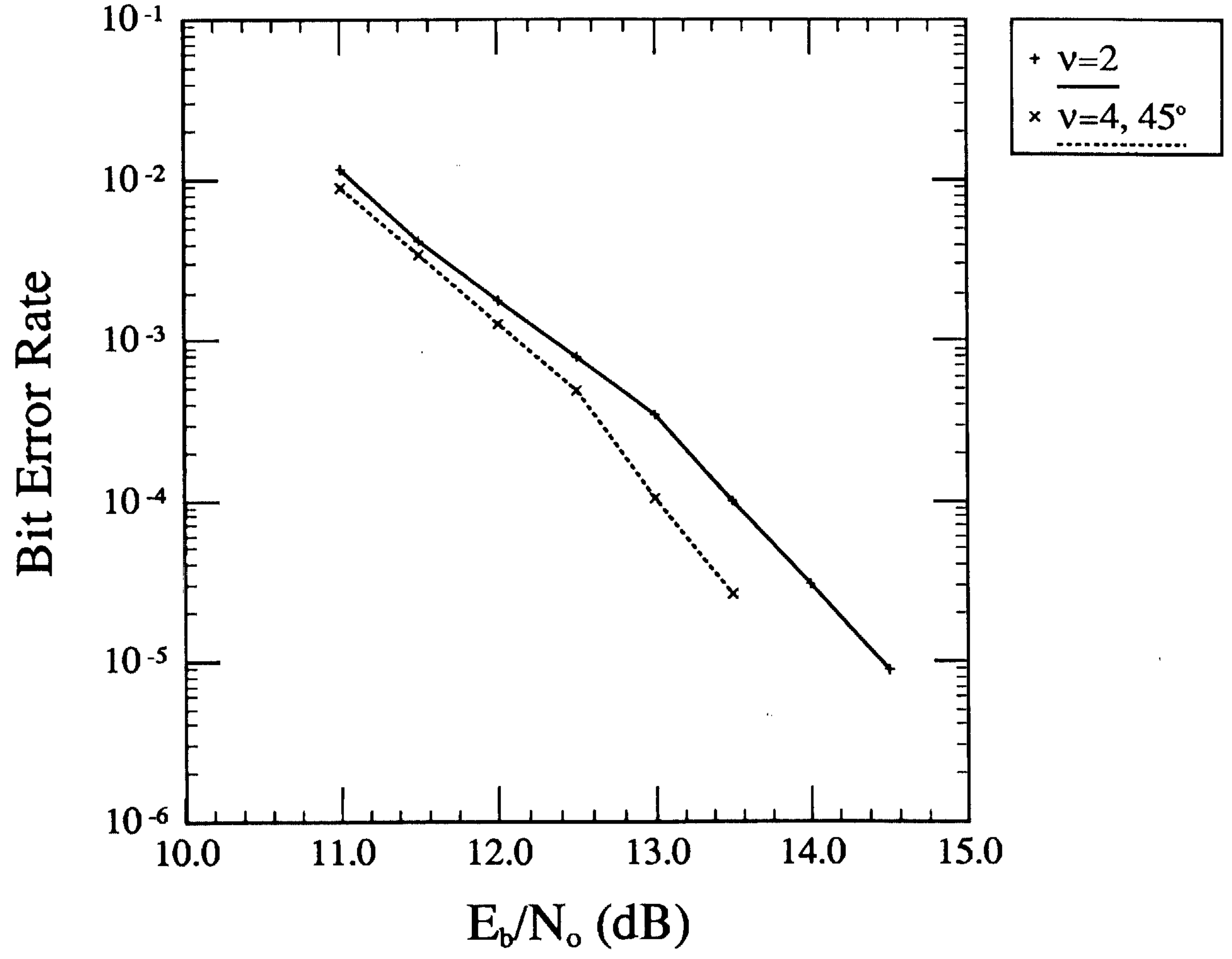


Figure A.24: 3x16PSK, R=3.67 bits/T Simulation Results (Table 21a)

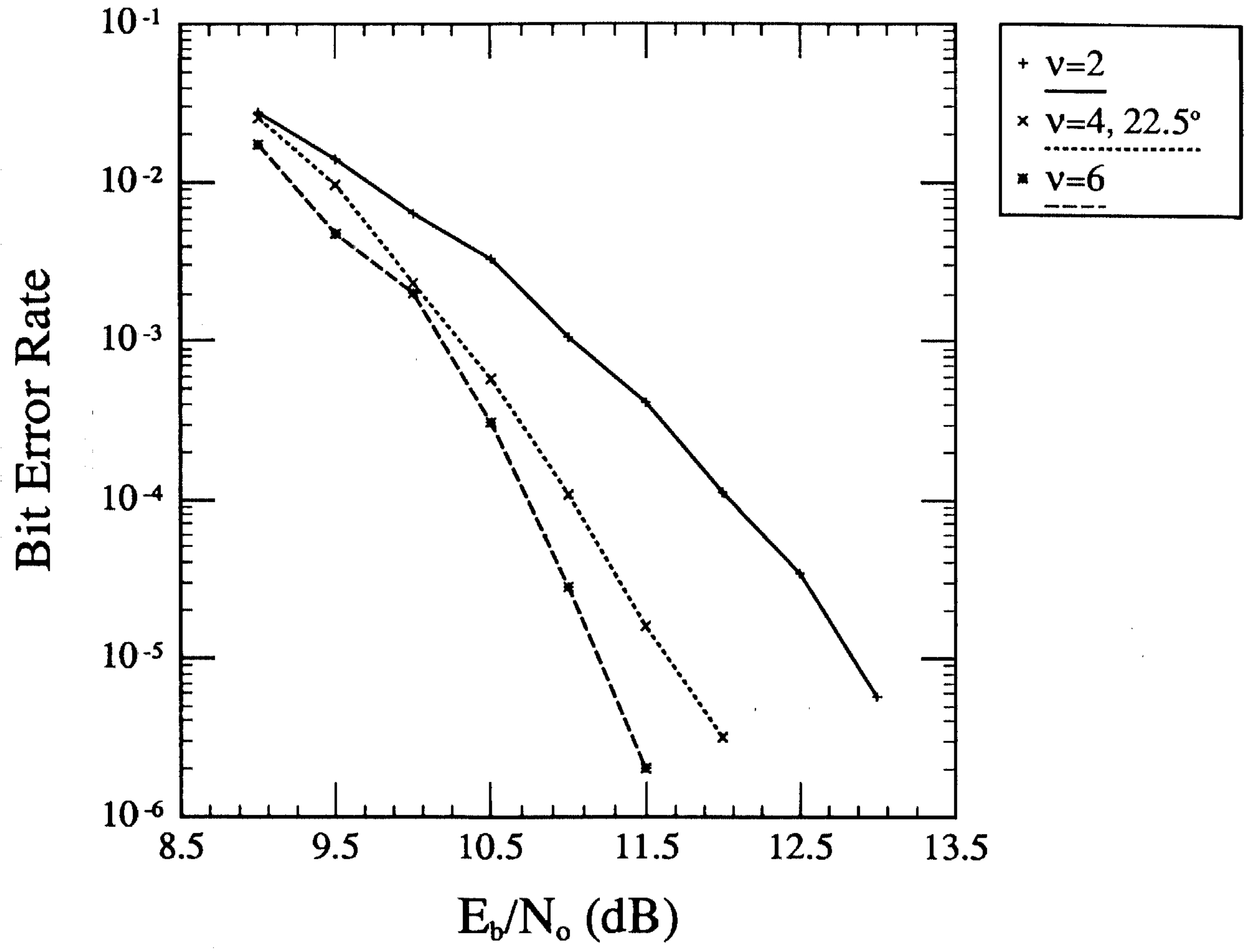


Figure A.25: 3x16PSK, R=3.33 bits/T Simulation Results (Table 21b)

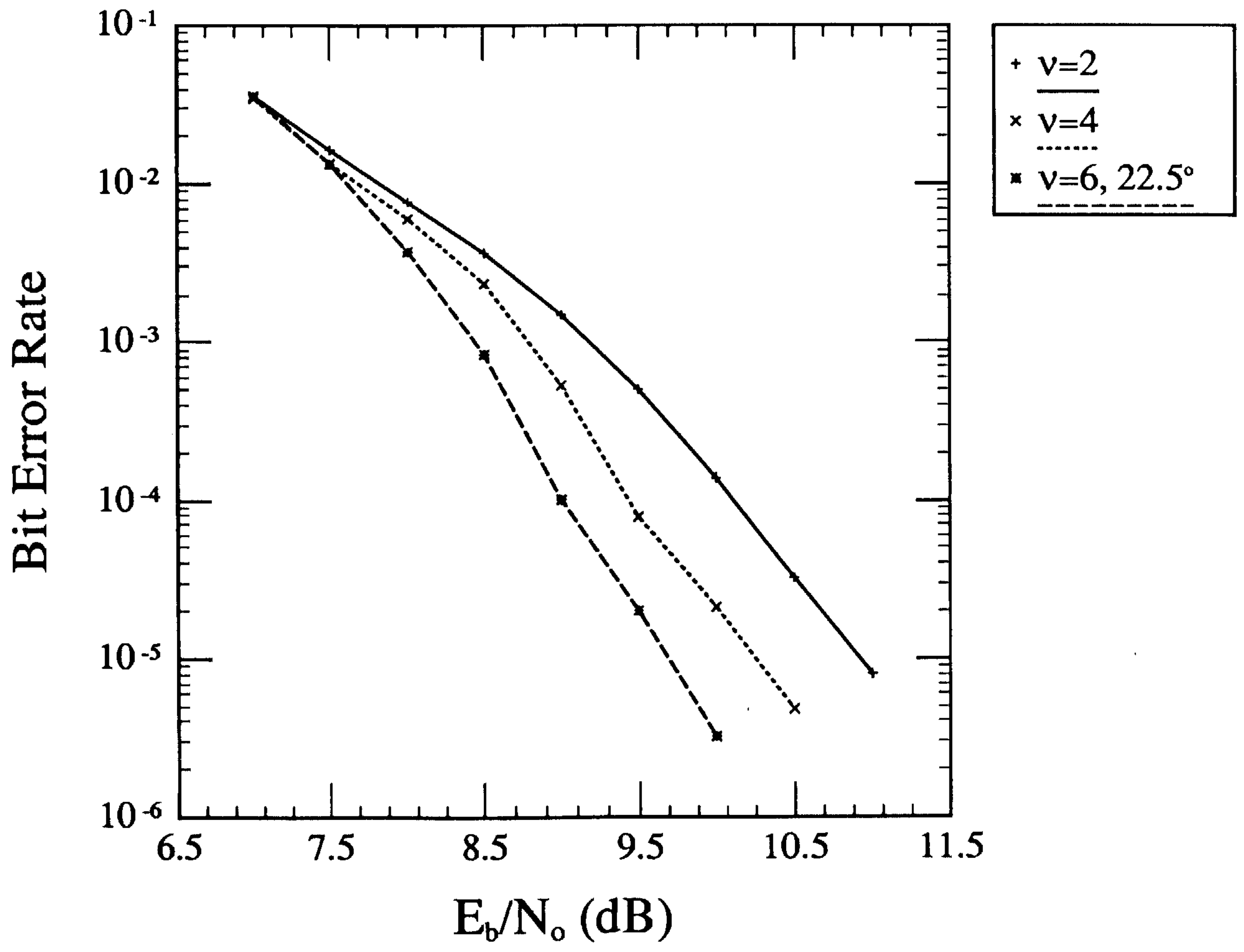


Figure A.26: 3x16PSK, R=3.00 bits/T Simulation Results (Table 21c)